RM0400
Reference manual

SPC572Lx - 32-bit Power Architecture® based MCU for automotive powertrain applications

Overview

This document defines the functionality of the SPC572Lx microcontroller for use by software and hardware developers. The SPC572Lx is built on Power Architecture® technology and integrates technologies that are important for today's and tomorrow's automotive applications, including automotive powertrain controller applications for four-cylinder gasoline and diesel engines, chassis control applications, transmission control applications, steering and braking applications, as well as low-end hybrid applications and safety applications that 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 SPC572Lx 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.

Document organization

This document includes chapters that are divided into parts:

- Part I includes chapters that describe the device as a whole or provide device-specific information.
- Part II includes chapters that describe the functionality of the individual modules on the device.

Contents

1 Preface 44

1.1 Overview 44

1.2 Audience 44

1.3 Document organization 44

1.4 Register conventions 44

1.5 Acronyms and abbreviations 45

1.6 Reference documents 45

2 Introduction 46

2.1 SPC572Lx microcontroller 46

2.1.1 Target applications 46

2.1.2 Overall architecture 46

2.1.3 Core features 46

2.1.4 Memory hierarchy 46

2.2 Features summary 47

2.3 Feature list 48

2.4 Packaging 49

2.5 Software debug and calibration 49

2.6 Block diagrams 50

3 Embedded memories 52

3.1 Overview 52

3.2 SRAM 52

3.2.1 System SRAM 52

3.2.2 Overlay SRAM 52

3.2.3 Processor core local SRAM 53

3.3 Embedded flash memory 53

3.3.1 Flash memory controller 54

3.3.2 Flash memory array 54

3.4 Security features 56

4 Signal description 57

4.1 Production packages 57

4.2 Package pinouts . . . 57
4.3 Pin descriptions . . . 57
4.3.1 Pin startup and reset states . . . 57
4.3.2 Power supply and reference voltage pins . . . 58
4.3.3 System pins . . . 58
4.3.4 LVDS pins . . . 59
4.3.5 Generic pins . . . 60

5 Memory map . . . 61

6 Device configuration . . . 69
6.1 Introduction . . . 69
6.2 Core module . . . 69
6.2.1 Core reset settings . . . 69
6.2.2 Special purpose register (SPR) summary . . . 71
6.3 System modules . . . 74
6.3.1 SIUL2 configuration . . . 74
6.3.2 Crossbar switch configuration . . . 74
6.3.3 System Memory Protection Unit (SMPU) configuration . . . 75
6.3.4 Peripheral bridge configuration . . . 77
6.3.5 Platform Configuration Module (PCM) . . . 78
6.3.6 Interrupt Controller (INTC) configuration . . . 79
6.3.7 DMA controller configuration . . . 98
6.3.8 DMACHMUX configuration . . . 99
6.4 Clocking . . . 102
6.5 Memories and memory interfaces . . . 102
6.5.1 Flash memory controller (PFLASH) configuration . . . 103
6.5.2 Decorated Storage Memory Controller (DSMC) . . . 104
6.5.3 ECC Error Reporting Module . . . 104
6.5.4 ECC Error Injection Module . . . 105
6.6 Analog modules . . . 105
6.6.1 Temperature sensor configuration . . . 106
6.6.2 Analog-to-Digital Converters (ADC) configuration . . . 106
6.7 Timers . . . 107
6.7.1 System Timer Module (STM) configuration . . . 108
6.7.2 Software Watchdog Timer (SWT) configuration . . . 108

6.7.3 PIT configuration . . . 109
6.7.4 GTM integration module configuration . . . 111
6.8 Communication interfaces . . . 111
6.8.1 FEC interface . . . 111
6.8.2 CAN nodes . . . 112
6.8.3 LFAST and SIPI . . . 112
6.8.4 DSPI configuration . . . 114
6.8.5 SENT Receiver (SRX) configuration . . . 119
6.8.6 LINFlexD – Configurations . . . 121
6.9 Reset and Boot modules . . . 123
6.9.1 BAF configuration . . . 123
6.9.2 System Status and Configuration Module (SSCM) configuration . . . 125
6.9.3 Safety modules . . . 125
6.10 Security modules . . . 130
6.10.1 Password and Device Security Module (PASS) configuration . . . 130
6.11 Debug and calibration architecture . . . 135

7 Reset and Boot . . . 137
7.1 Introduction . . . 137
7.1.1 Boot Assist Flash . . . 137
7.1.2 TEST flash memory block . . . 137
7.1.3 UTEST flash memory block . . . 137
7.1.4 Boot header . . . 138
7.2 Modules used in reset sequence . . . 138
7.2.1 Power Management Controller . . . 138
7.2.2 Reset Generation Module . . . 138
7.2.3 Mode Entry module . . . 139
7.2.4 System Status and Configuration Module . . . 139
7.3 Reset sequence . . . 139
7.3.1 Power-on and the Reset Generation Module . . . 140
7.3.2 Power-up phase: power stabilization . . . 141
7.3.3 PHASE0 Phase: analog supply initial configuration . . . 142
7.3.4 PHASE1: temporization and monitoring setup . . . 143
7.3.5 PHASE2: flash initial configuration . . . 143
7.3.6 PHASE3: device configuration . . . 143
7.3.7 IDLE Phase . . . 144

7.4 MC_RGM passes boot sequence control to the System Status and Control Module 145
7.4.1 Reset sequence flow based on initial device condition 145
7.4.2 Possible boot-up sequences 145
7.4.3 Sequence from MC_RGM IDLE to Boot CPU running code 145
7.4.4 Path from MC_RGM IDLE to serial boot mode 148
7.4.5 Path from MC_RGM IDLE with Core enabled to watchdog timer timeout 150

8 Device Configuration Format (DCF) Records 157
8.1 Introduction 157
8.2 DCF clients 157
8.3 DCF records 158
8.3.1 UTEST DCF records 161
8.4 DCF client table 161
8.4.1 DCF client list 162
8.4.2 Miscellaneous DCF registers 167

9 Power management 169
9.1 Overview 169
9.1.1 Power management framework 169
9.1.2 Power management supply description 170
9.1.3 SPC572Lx power management controller overview 171
9.2 Low power mode support 173
9.2.1 Low power support and STOP mode implementation 174
9.3 Flash power requirements 175
9.4 Device trimming 175
9.5 Supply monitoring (POR and LVDs) 175
9.5.1 Power-on reset (POR) 176
9.5.2 Behavior of device LVDs 176
9.5.3 Low Voltage Detection (LVD) 176
9.6 Power sequence 180
9.6.1 Power-up sequence 180
9.6.2 Power-down sequence 185
9.6.3 Brown-out management 185
9.6.4 Low voltage requirement during crank 186

10 Security . . . 187
10.1 Basic security . . . 187
10.2 Advanced security . . . 187
10.3 Detailed security information . . . 187

11 Calibration and Debug . . . 188
11.1 Core debug support . . . 188
11.2 Run control and memory access . . . 188
11.2.1 Debug and Calibration Interface (DCI) . . . 189
11.2.2 JTAG Data Communication (JDC) . . . 191
11.2.3 Sequence Processing Unit (SPU) . . . 191
11.3 Calibration interface . . . 203
11.3.1 JTAG Master (JTAGM) . . . 204
11.3.2 Development Trigger Semaphore (DTS) . . . 204
11.4 Debug over CAN . . . 204
11.5 Nexus Aurora . . . 205
11.5.1 Nexus Aurora register interface overview . . . 205
11.5.2 Nexus Aurora Router (NAR) . . . 206
11.6 Nexus clients . . . 208
11.6.1 e200z2 Nexus 3 . . . 208

12 Core e200z215An3 description . . . 209
12.1 Overview of the e200z215An3 core . . . 209
12.2 EFPU2 Floating-Point Unit features . . . 210
12.3 Register model . . . 210
12.4 Single-issue operation . . . 214
12.5 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR) . . . 214
12.6 Exceptions . . . 216
12.6.1 Exception Syndrome Register (ESR) . . . 217
12.6.2 Machine State Register (MSR) . . . 219
12.6.3 Machine Check Syndrome Register (MCSR) . . . 221
12.6.4 Interrupt Vector Prefix Registers (IVPR) . . . 223
12.6.5 Interrupt Definitions . . . 224

13 System Integration Unit Lite2 (SIUL2) . . . 233

13.1 Introduction . . . 233
13.1.1 Overview . . . 233
13.1.2 Features . . . 235
13.1.3 Register protection . . . 235
13.2 Memory map and register description . . . 236
13.2.1 Memory map . . . 236
13.2.2 Register descriptions . . . 237
13.3 Functional description . . . 254
13.3.1 General . . . 254
13.3.2 Pad control . . . 254
13.3.3 General purpose input or output pads (GPIO) . . . 255
13.3.4 External interrupts/DMA requests (EIRQ pins) . . . 256

14 Crossbar Switch (XBAR) . . . 259
14.1 Introduction . . . 259
14.1.1 Features . . . 259
14.1.2 Memory map and register definition . . . 259
14.2 Functional description . . . 264
14.2.1 General operation . . . 265
14.2.2 Register coherency . . . 265
14.2.3 Arbitration . . . 266
14.3 Initialization/application information . . . 267

15 Peripheral Bridge . . . 268
15.1 Introduction . . . 268
15.1.1 Features . . . 268
15.1.2 General operation . . . 268
15.2 Memory map and register definition . . . 268
15.2.1 Master Privilege Registers (AIPS_MPRA) . . . 269
15.2.2 Peripheral Access Control Registers (AIPS_PACR*x*) . . . 270
15.2.3 Off-Platform Peripheral Access Control Registers (AIPS_OPACR*x*) . . . 271
15.3 Functional description . . . 274
15.3.1 Access support . . . 274

16 System Memory Protection Unit (SMPU) . . . 275
16.1 Overview . . . 275

16.2 Block diagram . . . 275
16.3 Features . . . 277
16.4 Memory map and register definition . . . 277
16.4.1 Memory map . . . 277
16.4.2 Register descriptions . . . 281
16.5 Functional description . . . 290
16.5.1 Access evaluation macro . . . 290
16.5.2 Putting it all together and error terminations . . . 292
16.6 Initialization information . . . 292
16.7 Application information . . . 292

17 Intelligent AHB Gasket (IAHBG) . . . 295
17.1 Introduction . . . 295
17.2 Timing diagrams . . . 295
17.2.1 1:1 . . . 295
17.2.2 2:1 . . . 297
17.2.3 1:2 . . . 301
17.2.4 64->32 Interface . . . 305
17.2.5 32->64 Interface . . . 305

18 Interrupt Controller (INTC) . . . 307
18.1 Introduction . . . 307
18.2 Block diagram . . . 307
18.3 Features . . . 309
18.4 Modes of operation . . . 309
18.4.1 Software vector mode . . . 309
18.4.2 Hardware vector mode . . . 310
18.5 Memory map and register definition . . . 310
18.5.1 Memory map . . . 310
18.5.2 Register descriptions . . . 311
18.6 Functional description . . . 317
18.6.1 Interrupt request sources . . . 317
18.6.2 Priority management . . . 318
18.6.3 Handshaking with processor . . . 319
18.7 Initialization/application information . . . 322
18.7.1 Initialization flow . . . 322

18.7.2 Interrupt exception handler . . . 323
18.7.3 ISR, RTOS, and task hierarchy . . . 325
18.7.4 Order of execution . . . 325
18.7.5 Priority ceiling protocol . . . 327
18.7.6 Selecting priorities according to the request rates and deadlines . . . 329
18.7.7 Software-settable interrupt requests . . . 330
18.7.8 Lowering priority within an ISR . . . 330
18.7.9 Negating an interrupt request outside of its ISR . . . 331
18.7.10 Examining LIFO contents . . . 331
18.8 Interrupt sources . . . 332

19 Enhanced Direct Memory Access (eDMA) . . . 333
19.1 Introduction . . . 333
19.1.1 Features . . . 333
19.2 Modes of operation . . . 334
19.2.1 Normal mode . . . 334
19.2.2 Debug mode . . . 334
19.2.3 Wait mode . . . 335
19.3 Memory map/register definition . . . 335
19.3.1 Control Register (DMA_CR) . . . 347
19.3.2 Error Status Register (DMA_ES) . . . 349
19.3.3 Enable Request Register Low (DMA_ERQL) . . . 350
19.3.4 Enable Error Interrupt Register Low (DMA_EEIL) . . . 352
19.3.5 Set Enable Request Register (DMA_SERQ) . . . 354
19.3.6 Clear Enable Request Register (DMA_CERQ) . . . 355
19.3.7 Set Enable Error Interrupt Register (DMA_SEEI) . . . 355
19.3.8 Clear Enable Error Interrupt Register (DMA_CEEI) . . . 356
19.3.9 Clear Interrupt Request Register (DMA_CINT) . . . 357
19.3.10 Clear Error Register (DMA_CERR) . . . 357
19.3.11 Set START Bit Register (DMA_SSRT) . . . 358
19.3.12 Clear DONE Status Bit Register (DMA_CDNE) . . . 358
19.3.13 Interrupt Request Register Low (DMA_INTL) . . . 359
19.3.14 Error Register Low (DMA_ERRL) . . . 361
19.3.15 Hardware Request Status Register Low (DMA_HRSL) . . . 363
19.3.16 Channel *n* Priority Register (DMA_DCHPRI*n*) . . . 365
19.3.17 Channel *n* Master ID Register (DMA_DCHMID*n*) . . . 365
19.3.18 TCD Source Address (DMA_TCD*n*_SADDR) . . . 366

19.3.19 TCD Transfer Attributes (DMA_TCD*n*_ATTR) 367
19.3.20 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF) 368
19.3.21 TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD*n*_NBYTES_MLNO) 369
19.3.22 TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO) 369
19.3.23 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES) 370
19.3.24 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST) 371
19.3.25 TCD Destination Address (DMA_TCD*n*_DADDR) 372
19.3.26 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES) 372
19.3.27 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO) 373
19.3.28 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF) 374
19.3.29 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA) 375
19.3.30 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES) 375
19.3.31 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO) 376
19.3.32 TCD Control and Status (DMA_TCD*n*_CSR) 377
19.4 Functional description 379
19.4.1 eDMA microarchitecture 379
19.4.2 eDMA basic data flow 380
19.4.3 Error reporting and handling 383
19.4.4 Channel preemption 385
19.4.5 eDMA performance 385
19.5 Initialization/application information 388
19.5.1 eDMA initialization 388
19.5.2 DMA programming errors 390
19.5.3 DMA Arbitration mode considerations 390
19.5.4 DMA transfer 391
19.5.5 eDMA TCD*n* status monitoring 394
19.5.6 Channel linking 396
19.5.7 Dynamic programming 397

20 Direct Memory Access Multiplexer (DMAMUX) 400
20.1 Introduction 400

20.1.1 Overview . . . 400
20.1.2 Features . . . 401
20.1.3 Modes of operation . . . 401
20.2 External signal description . . . 401
20.3 Memory map/register definition . . . 401
20.3.1 Channel Configuration register (DMAMUX_CHCFGn) . . . 402
20.4 Functional description . . . 403
20.4.1 DMA channels with periodic triggering capability . . . 403
20.4.2 DMA channels with no triggering capability . . . 405
20.4.3 Always-enabled DMA sources . . . 405
20.5 Initialization/application information . . . 406
20.5.1 Reset . . . 406
20.5.2 Enabling and configuring sources . . . 406

21 Clocking . . . 411
21.1 Introduction . . . 411
21.2 Clock generation . . . 411
21.2.1 MC_CGM registers . . . 413
21.3 System clock frequency limitations . . . 413
21.3.1 JTAG frequencies . . . 414
21.4 Default clock configuration . . . 414
21.5 Clock sources . . . 415
21.5.1 PLL . . . 415
21.5.2 External oscillator (XOSC) . . . 417
21.5.3 16 MHz internal RC oscillator (IRCOSC) . . . 418
21.6 Peripheral clocks . . . 419
21.6.1 LFAST clocking . . . 420
21.6.2 Ethernet clocking . . . 422
21.6.3 M_CAN clocking . . . 422
21.6.4 Sigma-Delta ADC clocking . . . 423
21.6.5 Successive Approximation Register ADC clocking . . . 423
21.6.6 System Clock/GTM output clock/Ethernet clock pin muxing . . . 423
21.7 Clock monitoring . . . 424
21.7.1 CMU configuration . . . 424
21.7.2 PLL0 monitor . . . 426
21.7.3 External oscillator (XOSC) monitor . . . 426

21.7.4 Internal RC oscillator (IRCOSC) monitor 426
21.8 Loss of system clock behavior 427
21.8.1 Loss of PLL/XOSC clock 427
21.8.2 Loss of IRCOSC clock 427
21.9 Progressive clock switching 427

22 PLL Digital Interface (PLLDIG) 428
22.1 Introduction 428
22.2 Block Diagram 428
22.3 Features 428
22.4 Modes of operation 428
22.4.1 Normal mode with reference, PLL enabled 428
22.5 Memory map and register definition 429
22.5.1 Memory map 429
22.5.2 Register descriptions 429
22.5.3 Register classification for safety requirements 433
22.6 Functional description 433
22.6.1 Input clock frequency 433
22.6.2 Clock configuration 434
22.6.3 Loss of lock 435
22.7 Initialization information 435

23 Clock Monitor Unit (CMU) 436
23.1 Introduction 436
23.1.1 Main features 436
23.2 Block diagram 436
23.3 Signals 437
23.4 Memory map and register definition 438
23.4.1 Register descriptions 438
23.5 Functional description 443
23.5.1 Frequency meter 443
23.5.2 CLKMN0_RMT supervisor 443
23.5.3 CLKMN1 supervisor 443

24 Clock Generation Module (MC_CGM) 445
24.1 Introduction 445

24.1.1 Overview . . . 445
24.1.2 Features . . . 447
24.2 External signal description . . . 447
24.3 Memory map and register definition . . . 447
24.3.1 Register descriptions . . . 449
24.4 Functional Description . . . 481
24.4.1 System Clock Generation . . . 481
24.4.2 Auxiliary Clock Generation . . . 486
24.4.3 Dividers Functional Description . . . 491

25 OSC Digital Interface (XOSC) . . . 494
25.1 Introduction . . . 494
25.2 Functional description . . . 494
25.2.1 Oscillator power-down control and status . . . 494
25.2.2 Oscillator startup delay . . . 495
25.2.3 Oscillator clock available interrupt . . . 495
25.2.4 Oscillator bypass mode . . . 495
25.3 Memory map and register definition . . . 495
25.3.1 Register descriptions . . . 496

26 IRCOSC digital interface . . . 498
26.1 Introduction . . . 498
26.2 Functional description . . . 498
26.3 Memory map and register definition . . . 498
26.3.1 Register descriptions . . . 499

27 RAM controller (PRAM) . . . 502
27.1 Introduction . . . 502
27.2 Memory map and register definition . . . 503
27.2.1 Platform RAM Configuration Register 1 (PRAMC_PRCR1) . . . 503
27.3 SRAM ECC mechanism . . . 504
27.4 Functional description . . . 505
27.4.1 Read / Write introduction . . . 505
27.4.2 Initialization/application information . . . 506

28 Flash memory controller (PFLASH Controller) . . . 507

28.1 Introduction . . . 507
28.2 Features . . . 507
28.3 Block diagrams . . . 507
28.4 Flash memory controller memory map . . . 510
28.4.1 Overview . . . 510
28.5 Functional description . . . 531
28.5.1 Basic interface protocol . . . 531
28.5.2 Access protections . . . 531
28.5.3 Read cycles - buffer miss . . . 532
28.5.4 Read cycles – buffer hit . . . 532
28.5.5 Error termination . . . 532
28.5.6 Flash error response operation . . . 532
28.5.7 Censorship . . . 533
28.5.8 Line read buffers and prefetch operation . . . 533
28.5.9 Instruction/Data prefetch triggering . . . 534
28.5.10 Per-Master prefetch triggering . . . 534
28.5.11 Buffer allocation . . . 534
28.5.12 PFlash calibration remap support . . . 534
28.5.13 Safety considerations . . . 539
28.5.14 Array integrity considerations . . . 542

29 Embedded Flash Memory (MP55) . . . 543
29.1 Introduction . . . 543
29.1.1 Overview . . . 543
29.1.2 Features . . . 545
29.1.3 Modes of operation . . . 545
29.2 Flash memory map and description . . . 546
29.2.1 Flash array memory map . . . 547
29.3 Register memory maps and descriptions . . . 549
29.3.1 Register memory maps . . . 550
29.3.2 User register descriptions . . . 551
29.4 Functional description . . . 581
29.4.1 Reset . . . 581
29.4.2 Power-down mode (Disable mode) . . . 581
29.4.3 Low-power mode (Sleep mode) . . . 582
29.4.4 Read mode . . . 583

29.4.5 Modify mode 585
29.4.6 Protection strategy 595

30 Decorated Storage Memory Controller (DSMC) 597
30.1 Introduction 597
30.2 DSMC Block Diagram 598
30.3 Decorated Stores: st[b,h,w]d{cb}x rS,rB,rA 600
30.3.1 Bit Field Insert (BFINS) into an 8-, 16- or 32-bit Memory Container 600
30.4 DSMC Timing Diagram 605
30.5 DSMC Instantiations 606

31 Flash Memory Programming and Configuration 607
31.1 Selection of flash memory blocks for erase 608
31.2 Non-secure write protection 610
31.3 Secure write protection 611
31.3.1 Implementing secure write protection 612
31.3.2 Overriding secure write protection 615
31.4 Secure read protection 617
31.4.1 Implementing secure read protection 618
31.4.2 Overriding secure read protection 620
31.5 Debug port enable/disable 621
31.6 Implementing OTP 621
31.7 Implementing test mode disable 621
31.7.1 Unconditional test mode disable seal 622
31.7.2 Passcode-protected test mode disable seal 622
31.7.3 Selecting flash memory blocks for test mode disable seal 623
31.8 Security configuration planning 623
31.8.1 Creating password groups 623
31.8.2 Planning secure write protection 624
31.8.3 Planning secure read protection 624
31.8.4 Planning debug port enable/disable 624
31.8.5 Planning OTP flash memory block assignment 624
31.8.6 Planning factory test mode disable 624

32 Error Reporting Module (ERM) 625

32.1 Introduction . . . 625
32.1.1 Overview . . . 625
32.1.2 Features . . . 625
32.2 Memory map and register definition . . . 625
32.2.1 Memory map . . . 625
32.2.2 Register descriptions . . . 626
32.3 Functional description . . . 633
32.3.1 Single-bit correction events . . . 633
32.3.2 Non-correctable error events . . . 633

33 Error Injection Module (EIM) . . . 634
33.1 Introduction . . . 634
33.1.1 Overview . . . 634
33.1.2 Features . . . 635
33.2 Memory map and register definition . . . 635
33.2.1 Memory map . . . 635
33.2.2 Register descriptions . . . 636
33.3 Functional description . . . 641

34 Analog-to-digital converter (ADC) configuration . . . 643
34.1 ADC overview . . . 643
34.1.1 ADC subsystem block diagram . . . 643
34.1.2 Analog input pin multiplexing . . . 644
34.2 Configuration of ADC modules . . . 645
34.2.1 Sigma-Delta Analog-to-Digital Converter (SDADC) . . . 645
34.2.2 Successive Approximation Register Analog-to-Digital Converter (SARADC) . . . 651

35 Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface . . . 683
35.1 Introduction . . . 683
35.2 Overview . . . 683
35.3 Features . . . 685
35.4 Modes of operation . . . 685
35.4.1 Differential input mode . . . 685
35.4.2 Single-ended input mode . . . 685

35.4.3 External modulator mode . . . 685
35.5 External signal description . . . 686
35.5.1 Detailed signal descriptions . . . 686
35.6 Memory map and register descriptions . . . 687
35.6.1 Memory map . . . 687
35.6.2 Register descriptions . . . 687
35.7 Functional description . . . 699
35.7.1 Differential input mode . . . 700
35.7.2 Single-ended input mode . . . 700
35.7.3 External modulator mode . . . 700
35.7.4 Low consumption mode . . . 700
35.7.5 Analog input multiplexing and bias support . . . 700
35.7.6 Programmable gain and decimation rate . . . 701
35.7.7 High-pass filter support . . . 701
35.7.8 Data conversion . . . 701
35.7.9 Hardware triggering . . . 702
35.7.10 Interrupt/DMA request support . . . 703
35.7.11 Gain calibration support . . . 703
35.7.12 Offset calibration support . . . 704
35.8 Initialization information . . . 704
35.8.1 Data conversion step . . . 705

36 Successive Approximation Register Analog-to-Digital Converter (SARADC) . . . 706
36.1 Introduction . . . 706
36.2 Overview . . . 706
36.3 Feature description . . . 708
36.3.1 Main features . . . 708
36.4 Functional description . . . 708
36.4.1 Normal channel conversion . . . 708
36.4.2 Injected channel conversion . . . 710
36.4.3 Abort conversion . . . 711
36.4.4 Analog conversion timings and reference selection . . . 712
36.4.5 Test channel connection with internal analog channel . . . 713
36.4.6 External channel mapping to internal analog channel . . . 714
36.4.7 Programmable analog watchdog . . . 714

36.4.8 DMA functionality . . . 715
36.4.9 Interrupts . . . 715
36.4.10 External decode signals selection and delay . . . 716
36.4.11 Power down mode . . . 717
36.4.12 Stop Mode . . . 717
36.5 Memory map and registers . . . 717
36.5.1 Register descriptions . . . 720
36.6 Start of conversion pulse delay . . . 763
36.7 Initialization Information . . . 765

37 Decimation Filter . . . 767
37.1 Introduction . . . 767
37.1.1 Overview . . . 767
37.1.2 Features . . . 769
37.1.3 Modes of operation . . . 769
37.2 External signal description . . . 770
37.2.1 Integrator enable signal . . . 770
37.2.2 Integrator halt signal . . . 770
37.2.3 Integrator reset signal . . . 770
37.2.4 Integrator output request signal . . . 770
37.3 Memory map and register definition . . . 770
37.3.1 Decimation filter device memory map . . . 771
37.3.2 Decimation filter register descriptions . . . 772
37.4 Functional description . . . 789
37.4.1 Overview . . . 789
37.4.2 Data interface sub-block description . . . 790
37.4.3 Input buffer description . . . 790
37.4.4 Output buffer description . . . 791
37.4.5 Bypass configuration description . . . 791
37.4.6 IIR and FIR Filter . . . 792
37.4.7 Filter prefill control description . . . 795
37.4.8 Flush command description . . . 795
37.4.9 Soft reset command description . . . 796
37.4.10 Interrupts requests description . . . 797
37.4.11 DMA requests description . . . 798
37.4.12 Freeze mode description . . . 798

37.4.13 Integrator . . . 799
37.5 Initialization Information . . . 802
37.5.1 Initialization Procedure . . . 802
37.6 Filter example simulation . . . 803
37.6.1 Coefficients calculation . . . 803
37.6.2 Input data calculation . . . 804
37.6.3 Filter results . . . 804

38 Temperature Sensor . . . 805
38.1 Introduction . . . 805
38.1.1 Signal Descriptions . . . 805
38.2 Functional Description . . . 805
38.2.1 Temperature threshold detection (digital output generation) . . . 805
38.2.2 Linear temperature sensor (analog output generation) . . . 806
38.3 Temperature formula . . . 807
38.4 Calculating device temperature . . . 808

39 System Timer Module (STM) . . . 809
39.1 Introduction . . . 809
39.1.1 Overview . . . 809
39.1.2 Features . . . 809
39.1.3 Modes of operation . . . 809
39.2 External signal description . . . 809
39.3 Memory map and register definition . . . 809
39.3.1 Memory map . . . 809
39.3.2 Register descriptions . . . 810
39.4 Functional description . . . 813

40 Software Watchdog Timer (SWT) . . . 814
40.1 Introduction . . . 814
40.1.1 Overview . . . 814
40.1.2 Features . . . 814
40.1.3 Modes of operation . . . 814
40.2 External signal description . . . 814
40.3 Memory map and register definition . . . 814
40.3.1 Register descriptions . . . 815

40.4 Functional description 820

41 Periodic Interrupt Timer (PIT) 822

41.1 Introduction 822

41.1.1 Block diagram 822

41.1.2 Features 823

41.2 Signal description 823

41.2.1 Memory map/register description 823

41.3 Functional description 829

41.3.1 General 829

41.3.2 Interrupts 830

41.3.3 Chained timers 830

41.4 Initialization and application information 830

41.5 Example configuration for chained timers 831

41.6 Example Configuration for the Lifetime Timer 832

42 GTM Development Interface (GTMDI) 833

42.1 About this module 833

42.2 Introduction 833

42.3 Overview 834

42.3.1 Features 838

42.3.2 Modes of operation 839

42.3.3 DATA port 839

42.4 External signal description 839

42.4.1 Event in (EVTI) 840

42.4.2 Event out EVTO[1:0] 840

42.4.3 Test clock input (TCK) 840

42.4.4 Test data input (TDI) 840

42.4.5 Test data output (TDO) 840

42.4.6 Test Mode Select (TMS) 841

42.5 Register definition 841

42.5.1 Register descriptions 843

42.5.2 Unimplemented registers 905

42.6 Functional description 905

42.6.1 Debug enable logic 906

42.6.2 TIM-TOM-ATOM selection and control logic 906

42.6.3 GTMDI reset configuration . . . 907
42.6.4 Message Data Bus —interface with NAR module . . . 908
42.6.5 IEEE 1149.1 (JTAG) input port . . . 912
42.6.6 Nexus Class 1 development support . . . 917
42.6.7 Debug status . . . 917
42.6.8 Data trace . . . 917
42.6.9 Fetch trace . . . 920
42.6.10 Watchpoint trace . . . 922
42.7 Initialization/application information . . . 922
42.7.1 Accessing GTMDI tool-mapped registers . . . 922

43 GTM101 Integration (GTMINT) Module . . . 924
43.1 Introduction . . . 924
43.1.1 Features . . . 924
43.1.2 Modes of operation . . . 926
43.1.3 Block diagram . . . 926
43.2 Memory map and register definition . . . 927
43.2.1 Module memory map . . . 927
43.2.2 Register descriptions . . . 947
43.3 Functional description . . . 950
43.3.1 GTM-IP Timer . . . 951
43.3.2 GTMDI Debug Interface . . . 952
43.3.3 Error Detection on GTM Memories Accesses . . . 953
43.3.4 Stop Mode Description . . . 955
43.3.5 Debug / Halt Mode Description . . . 956
43.3.6 GTM-IP Timer Software Reset Description . . . 956
43.3.7 AEI Interface Software Reset Description . . . 956
43.3.8 Interrupt and DMA Support Logic . . . 957
43.4 Initialization/Application Information . . . 958

44 CAN Subsystem . . . 959
44.1 Introduction . . . 959
44.2 Features . . . 960
44.3 Modular CAN cores . . . 961
44.3.1 Features . . . 962
44.3.2 Block diagram . . . 963

44.3.3 Dual clock sources . . . 964
44.3.4 Dual interrupt lines . . . 964
44.3.5 Memory map and register descriptions . . . 964
44.3.6 Message RAM . . . 1009
44.3.7 M_CAN functional description . . . 1017
44.3.8 Timestamp generation . . . 1025
44.3.9 Timeout counter . . . 1026
44.3.10 Rx handling . . . 1026
44.3.11 Dedicated Rx Buffers . . . 1030
44.3.12 Debug on CAN Support . . . 1031
44.3.13 Tx handling . . . 1032
44.3.14 FIFO acknowledge handling . . . 1035
44.3.15 Interface to DMA Controller . . . 1036
44.3.16 Clock Calibration on M_CAN . . . 1036
44.4 CAN RAM arbiter . . . 1049
44.4.1 Features . . . 1049
44.4.2 Functional overview using examples . . . 1050
44.5 SRAM interface and memory organization . . . 1050
44.5.1 Shared memory . . . 1050
44.5.2 ECC controller . . . 1051
44.5.3 Transmit message buffer locking mechanism . . . 1051
44.6 CAN nodes 1 and 2 I/Os sharing . . . 1052
44.7 External signal description . . . 1052
44.8 Memory map and register description . . . 1053
44.8.1 Shared memory map . . . 1053

45 Serial Interprocessor Interface (SIPI) . . . 1055
45.1 Introduction . . . 1055
45.1.1 Scalability . . . 1055
45.2 Overview . . . 1055
45.3 SIPI block diagram . . . 1057
45.4 Feature description . . . 1057
45.4.1 Main features . . . 1058
45.4.2 Standard features . . . 1058
45.5 SIPI operation from reset . . . 1058
45.6 Functional description . . . 1058

45.6.1 External signals . . . 1058
45.6.2 Frame format . . . 1058
45.7 Transfer types . . . 1064
45.7.1 Read transfer . . . 1064
45.7.2 Register read answer transfer . . . 1065
45.7.3 Register Write transfer . . . 1066
45.7.4 Write Acknowledge transfer . . . 1069
45.7.5 ID request response . . . 1069
45.8 Transfer API and flow charts . . . 1070
45.9 DMA programming sequence . . . 1077
45.10 Modes of operation . . . 1077
45.10.1 Initialization mode (INIT) . . . 1077
45.10.2 Normal mode (NM) . . . 1077
45.10.3 Module disable (MD) . . . 1078
45.11 Errors . . . 1078
45.11.1 Timeout error . . . 1078
45.11.2 CRC error . . . 1078
45.11.3 Maximum count reached error . . . 1078
45.11.4 Transaction ID error . . . 1078
45.11.5 Acknowledge error . . . 1079
45.12 CRC calculation . . . 1079
45.13 Interrupt logic . . . 1079
45.14 SIPI control and status overview . . . 1080
45.15 Memory map and register description . . . 1081
45.15.1 Register descriptions . . . 1083

46 Deserial Serial Peripheral Interface (DSPI) . . . 1097

46.1 Introduction . . . 1097
46.1.1 Block diagram . . . 1097
46.1.2 Features . . . 1098
46.1.3 DSPI configurations . . . 1099
46.1.4 Modes of operation . . . 1101
46.2 DSPI signal description . . . 1102
46.2.1 PCS0/SS — Peripheral Chip Select/Slave Select . . . 1102
46.2.2 PCS1 – PCS3 — Peripheral Chip Selects 1 – 3 . . . 1102
46.2.3 PCS4 — Peripheral Chip Select 4 . . . 1102

46.2.4 PCS5/PCSS — Peripheral Chip Select 5/ Peripheral Chip Select Strobe . . . 1102
46.2.5 PCS[6] – PCS[7] — Peripheral Chip Selects 6 – 7 . . . 1103
46.2.6 SIN — Serial Input . . . 1103
46.2.7 SOUT — Serial Output . . . 1103
46.2.8 SCK — Serial Clock . . . 1103
46.3 Memory map/register definition . . . 1103
46.3.1 DSPI Module Configuration Register (DSPI_MCR) . . . 1106
46.3.2 DSPI Transfer Count Register (DSPI_TCR) . . . 1109
46.3.3 DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*) . . . 1110
46.3.4 DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR*n*_SLAVE) . . . 1115
46.3.5 DSPI Status Register (DSPI_SR) . . . 1116
46.3.6 DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER) . . . 1119
46.3.7 DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR) . . . 1122
46.3.8 DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE) . . . 1124
46.3.9 DSPI POP FIFO Register (DSPI_POPR) . . . 1125
46.3.10 DSPI Transmit FIFO Registers (DSPI_TXFRn) . . . 1126
46.3.11 DSPI Receive FIFO Registers (DSPI_RXFRn) . . . 1126
46.3.12 DSPI DSI Configuration Register 0 (DSPI_DSICR0) . . . 1127
46.3.13 DSPI DSI Serialization Data Register 0 (DSPI_SDR0) . . . 1129
46.3.14 DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0) . . . 1130
46.3.15 DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0) . . . 1131
46.3.16 DSPI DSI Deserialization Data Register 0 (DSPI_DDR0) . . . 1131
46.3.17 DSPI DSI Configuration Register 1 (DSPI_DSICR1) . . . 1132
46.3.18 DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0) . . . 1133
46.3.19 DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0) . . . 1134
46.3.20 DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0) . . . 1135
46.3.21 DSPI DSI Serialization Data Register 1 (DSPI_SDR1) . . . 1135
46.3.22 DSPI DSI Alternate Serialization Data Register 1 (DSPI_ASDR1) . . . 1136
46.3.23 DSPI DSI Transmit Comparison Register 1 (DSPI_COMPR1) . . . 1137
46.3.24 DSPI DSI Deserialization Data Register 1 (DSPI_DDR1) . . . 1137

46.3.25 DSPI DSI Serialization Source Select Register 1 (DSPI_SSR1) . . . 1138
46.3.26 DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1) . . . 1139
46.3.27 DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1) . . . 1139
46.3.28 DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*) . . . 1140
46.3.29 DSPI Status Register Extended (DSPI_SREX) . . . 1141
46.4 Register classification for safety . . . 1142
46.5 Functional description . . . 1142
46.5.1 Start and Stop of DSPI transfers . . . 1143
46.5.2 Serial Peripheral Interface (SPI) configuration . . . 1144
46.5.3 Deserial Serial Interface (DSI) configuration . . . 1148
46.5.4 Combined Serial Interface (CSI) configuration . . . 1151
46.5.5 DSPI baud rate and clock delay generation . . . 1153
46.5.6 Transfer formats . . . 1156
46.5.7 Continuous Serial Communications Clock . . . 1165
46.5.8 Slave mode operation constraints . . . 1167
46.5.9 Timed Serial Bus (TSB) . . . 1167
46.5.10 Interleaved TSB (ITSB) Mode . . . 1170
46.5.11 Parity generation and check . . . 1173
46.5.12 Interrupts/DMA requests . . . 1174
46.5.13 Power saving features . . . 1177
46.6 Initialization/application information . . . 1178
46.6.1 Managing DSPI queues . . . 1178
46.6.2 Switching master and slave mode . . . 1178
46.6.3 Initializing DSPI in Master/Slave Modes . . . 1179
46.6.4 Baud rate settings . . . 1179
46.6.5 Delay settings . . . 1180
46.6.6 Calculation of FIFO pointer addresses . . . 1181

47 LVDS Fast Asynchronous Serial Transmission – Interprocessor Communications . . . 1184

47.1 Introduction . . . 1184
47.2 Block diagram . . . 1184
47.3 External signals . . . 1184
47.3.1 LFAST operating data rates . . . 1185

47.4 LFAST frame structure 1185
47.5 Features 1188
47.6 Memory map and register definition 1188
47.6.1 Memory map 1188
47.6.2 Register descriptions 1191
47.6.3 Register safety classification requirements 1218
47.7 Functional description 1218
47.7.1 Startup procedure 1218
47.7.2 Line Receiver (LR) 1221
47.7.3 Transmit Controller 1229
47.7.4 CTS mode support 1233
47.7.5 Frames supported 1234
47.7.6 Frame flow 1234
47.7.7 Test and Debug Support 1240
47.7.8 Interrupts 1247
47.8 Packet memory 1249
47.9 Resets 1250
47.10 Clocks 1250
47.10.1 Clocking strategy 1250
47.10.2 Slow speed clock 1251
47.10.3 Rx controller clocks 1254
47.10.4 Tx controller clocks 1255

48 Fast Ethernet Controller (FEC) 1257
48.1 Introduction 1257
48.1.1 Block diagram 1257
48.1.2 Features 1260
48.2 Modes of operation 1260
48.2.1 Full and half duplex operation 1260
48.2.2 Interface options 1261
48.2.3 Address recognition options 1261
48.2.4 Internal loopback 1261
48.3 External signal description 1262
48.4 Memory map and register definition 1263
48.4.1 Interrupt Event Register (EIR) 1266
48.4.2 Interrupt Mask Register (EIMR) 1268

48.4.3 Receive Descriptor Active Register (RDAR) 1269
48.4.4 Transmit Descriptor Active Register (TDAR) 1270
48.4.5 Ethernet Control Register (ECR) 1271
48.4.6 MII Management Frame Register (MMFR) 1271
48.4.7 MII Speed Control Register (MSCR) 1273
48.4.8 MIB Control Register (MIBC) 1274
48.4.9 Receive Control Register (RCR) 1274
48.4.10 Transmit Control Register (TCR) 1276
48.4.11 Physical Address Lower Register (PALR) 1277
48.4.12 Physical Address Upper Register (PAUR) 1278
48.4.13 Opcode/Pause Duration Register (OPD) 1278
48.4.14 Descriptor Individual Upper Address Register (IAUR) 1279
48.4.15 Descriptor Individual Lower Address Register (IALR) 1279
48.4.16 Descriptor Group Upper Address Register (GAUR) 1279
48.4.17 Descriptor Group Lower Address Register (GALR) 1280
48.4.18 Transmit FIFO Watermark Register (TFWR) 1280
48.4.19 FIFO Receive Bound Register (FRBR) 1281
48.4.20 FIFO Receive Start Register (FRSR) 1281
48.4.21 Receive Descriptor Ring Start Register (ERDSR) 1282
48.4.22 Transmit Buffer Descriptor Ring Start Register (ETSDR) 1282
48.4.23 Receive Buffer Size Register (EMRBR) 1283
48.4.24 Count of frames not counted correctly (RMON_T_DROP) 1284
48.4.25 RMON Tx packet count (RMON_T_PACKETS) 1284
48.4.26 RMON Tx broadcast packets (RMON_T_BC_PKT) 1284
48.4.27 RMON Tx multicast packets (RMON_T_MC_PKT) 1285
48.4.28 RMON Tx packets with CRC/align error (RMON_T_CRC_ALIGN) 1285
48.4.29 RMON Tx packets < 64 bytes, good CRC (RMON_T_UNDERSIZE) 1285
48.4.30 RMON Tx packets > MAX_FL bytes, good CRC (RMON_T_OVERSIZE) 1286
48.4.31 RMON Tx packets < 64 bytes, bad CRC (RMON_T_FRAG) 1286
48.4.32 RMON Tx packets > MAX_FL bytes, bad CRC (RMON_T_JAB) 1286
48.4.33 RMON Tx collision count (RMON_T_COL) 1287
48.4.34 RMON Tx 64 byte packets (RMON_T_P64) 1287
48.4.35 RMON Tx 65 to 127 byte packets (RMON_T_P65TO127) 1287
48.4.36 RMON Tx 128 to 255 byte packets (RMON_T_P128TO255) 1288

48.4.37 RMON Tx 256 to 511 byte packets (RMON_T_P256TO511) 1288
48.4.38 RMON Tx 512 to 1023 byte packets (RMON_T_P512TO1023) 1288
48.4.39 RMON Tx 1024 to 2047 byte packets (RMON_T_P1024TO2047) 1289
48.4.40 RMON Tx packets with > 2048 bytes (RMON_T_P_GTE2048) 1289
48.4.41 RMON Tx Octets (RMON_T_OCTETS) 1289
48.4.42 Count of transmitted frames not counted correctly (IEEE_T_DROP) 1290
48.4.43 Frames transmitted OK (IEEE_T_FRAME_OK) 1290
48.4.44 Frames transmitted with single collision (IEEE_T_1COL) 1290
48.4.45 Frames transmitted with multiple collisions (IEEE_T_MCOL) 1291
48.4.46 Frames transmitted after deferral delay (IEEE_T_DEF) 1291
48.4.47 Frames transmitted with late collision (IEEE_T_LCOL) 1291
48.4.48 Frames transmitted with excessive collisions (IEEE_T_EXCOL) . . . 1292
48.4.49 Frames transmitted with Tx FIFO underrun (IEEE_T_MACERR) . . . 1292
48.4.50 Frames transmitted with carrier sense error (IEEE_T_CSERR) 1292
48.4.51 Frames transmitted with SQE error (IEEE_T_SQE) 1293
48.4.52 Flow control pause frames transmitted (IEEE_T_FDXFC) 1293
48.4.53 Octet count for frames transmitted without error (IEEE_T_OCTETS_OK) 1293
48.4.54 Count of received frames not counted correctly (RMON_R_DROP) 1294
48.4.55 RMON Rx packet count (RMON_R_PACKETS) 1294
48.4.56 RMON Rx broadcast packets (RMON_R_BC_PKT) 1294
48.4.57 RMON Rx multicast packets (RMON_R_MC_PKT) 1295
48.4.58 RMON Rx packets with CRC/Align error (RMON_R_CRC_ALIGN) 1295
48.4.59 RMON Rx packets < 64 bytes, good CRC (RMON_R_UNDERSIZE) 1295
48.4.60 RMON Rx packets > MAX_FL bytes, good CRC (RMON_R_OVERSIZE) 1296
48.4.61 RMON Rx packets < 64 bytes, bad CRC (RMON_R_FRAG) 1296
48.4.62 RMON Rx packets > MAX_FL bytes, bad CRC (RMON_R_JAB) 1296
48.4.63 Reserved (RMON_R_RESVD_0) 1297
48.4.64 RMON Rx 64 byte packets (RMON_R_P64) 1297
48.4.65 RMON Rx 65 to 127 byte packets (RMON_R_P65TO127) 1297
48.4.66 RMON Rx 128 to 255 byte packets (RMON_R_P128TO255) 1298
48.4.67 RMON Rx 256 to 511 byte packets (RMON_R_P256TO511) 1298

48.4.68 RMON Rx 512 to 1023 byte packets (RMON_R_P512TO1023) 1298
48.4.69 RMON Rx 1024 to 2047 byte packets (RMON_R_P1024TO2047) 1299
48.4.70 RMON Rx packets with > 2048 bytes (RMON_R_P_GTE2048) 1299
48.4.71 RMON Rx octets (RMON_R_OCTETS) 1299
48.4.72 Count of received frames not counted correctly (IEEE_R_DROP) 1300
48.4.73 Frames received OK (IEEE_R_FRAME_OK) 1300
48.4.74 Frames received with CRC error (IEEE_R_CRC) 1300
48.4.75 Frames received with alignment error (IEEE_R_ALIGN) 1301
48.4.76 Receive FIFO overflow count (IEEE_R_MACERR) 1301
48.4.77 Flow control pause frames received (IEEE_R_FDXFC) 1301
48.4.78 Octet count for frames received without error (IEEE_R_OCTETS_OK) 1302
48.5 Functional description 1302
48.5.1 MII data frame 1302
48.5.2 MII management frame structure 1303
48.5.3 Buffer descriptors 1304
48.5.4 Initialization sequence 1309
48.5.5 User initialization (Prior to Setting ECR[ETHER_EN]) 1309
48.5.6 Microcontroller initialization 1310
48.5.7 User initialization (after setting ECR[ETHER_EN]) 1310
48.5.8 Network interface options 1310
48.5.9 FEC frame transmission 1312
48.5.10 FEC frame reception 1313
48.5.11 Ethernet address recognition 1314
48.5.12 Hash algorithm 1316
48.5.13 Full duplex flow control 1319
48.5.14 Inter-Packet Gap (IPG) time 1320
48.5.15 Collision managing 1320
48.5.16 MII internal and external loopback 1320
48.5.17 RMII loopback 1320
48.5.18 RMII echo 1321
48.5.19 Ethernet error-managing procedure 1321

49 SENT Receiver (SRX) 1323
49.1 Introduction 1323
49.1.1 Features 1324

49.1.2 Modes of operation . . . 1325
49.1.3 Block diagram . . . 1326
49.1.4 Design overview . . . 1327
49.2 External signal description . . . 1328
49.2.1 SENT_RX [(CH-1):0] . . . 1328
49.3 Memory map and register definition . . . 1328
49.3.1 Memory map . . . 1328
49.3.2 Register descriptions . . . 1330
49.3.3 Register Classification for Safety . . . 1360
49.4 Functional description . . . 1361
49.4.1 Initialization sequence . . . 1361
49.4.2 DMA read logic . . . 1362
49.4.3 Message reading via interrupts . . . 1366
49.4.4 Overflow behavior . . . 1367
49.4.5 Adjustment for variation in sensor (Tx) clock . . . 1368
49.4.6 Input programmable filter . . . 1368
49.4.7 Receiver diagnostics . . . 1369
49.4.8 Time stamp logic . . . 1371
49.4.9 Bus Idle Diagnostic . . . 1372
49.5 Clocks and resets . . . 1372
49.5.1 Clocking strategy . . . 1372
49.5.2 System bus clock requirements . . . 1373
49.5.3 High frequency receiver clock (protocol clock) requirements . . . 1374

50 LINFlexD . . . 1375
50.1 Introduction . . . 1375
50.2 Main features . . . 1377
50.2.1 LIN mode features . . . 1377
50.2.2 UART mode features . . . 1378
50.2.3 Standard features . . . 1378
50.3 Functional description . . . 1378
50.3.1 LIN protocol . . . 1378
50.3.2 LINFlexD features . . . 1380
50.3.3 Timer . . . 1396
50.3.4 UART mode . . . 1397
50.3.5 DMA interface . . . 1402

50.4 Memory map and register description 1420
50.4.1 Memory map 1420
50.4.2 Register description 1422
50.4.3 Register map 1457
50.5 Programming considerations 1461
50.5.1 Master node 1461
50.5.2 Slave node 1462
50.5.3 Extended frames 1467
50.5.4 Timeout 1467
50.5.5 UART mode 1468
50.5.6 Interrupts 1468

51 Reset Generation Module (MC_RGM) 1471
51.1 Introduction 1471
51.1.1 Overview 1471
51.1.2 Features 1473
51.1.3 Reset sources 1473
51.2 External signal description 1474
51.3 Memory map and register definition 1475
51.3.1 Register descriptions 1476
51.4 Functional description 1500
51.4.1 Reset state machine 1500
51.4.2 ‘Destructive’ resets 1503
51.4.3 External reset 1503
51.4.4 ‘Functional’ resets 1504
51.4.5 Alternate event generation 1505
51.4.6 ‘Functional’ reset escalation 1505
51.4.7 Individual peripheral resets 1506

52 Boot Assist Flash (BAF) 1507
52.1 Introduction 1507
52.1.1 BAF image header 1507
52.2 Functional description 1508
52.2.1 Boot modes 1508
52.2.2 Device initialization 1508
52.2.3 Flow of control 1509

52.2.4 Initialization of BAF DCF clients . . . 1509
52.2.5 Optionally perform a serial boot . . . 1515
52.2.6 Serial boot configuration . . . 1520
52.3 Resources . . . 1523

53 System Status and Configuration Module (SSCM) . . . 1524
53.1 Introduction . . . 1524
53.1.1 Overview . . . 1524
53.1.2 Glossary . . . 1524
53.1.3 Features . . . 1525
53.1.4 Modes of operation . . . 1525
53.2 External signal description . . . 1525
53.3 Memory map and register definition . . . 1525
53.3.1 Register descriptions . . . 1526
53.4 Functional description . . . 1529
53.4.1 DCF mechanism . . . 1529
53.4.2 ECC error monitoring . . . 1531
53.4.3 Boot mode functionality . . . 1531
53.4.4 BAF configuration . . . 1532
53.4.5 Life Cycle . . . 1532
53.5 Initialization and application information . . . 1533
53.5.1 Reset . . . 1533
53.6 Additional safety measures . . . 1533
53.6.1 Spurious reset protection . . . 1533

54 Power Management Controller digital interface (PMC_dig) . . . 1534
54.1 Introduction . . . 1534
54.2 Block diagram . . . 1534
54.3 Memory map and register definition . . . 1536
54.3.1 Register descriptions . . . 1537
54.4 Analog PMC interface . . . 1556
54.5 Temperature Sensor Interface Logic . . . 1556
54.6 Power On Reset (MC_RGM Phase Gates) . . . 1556
54.7 Flash interface (DCF client usage) . . . 1557

55 Power Control Unit (MC_PCU) . . . 1559

55.1 Introduction . . . 1559
55.1.1 Overview . . . 1559
55.1.2 Features . . . 1559
55.2 External signal description . . . 1560
55.3 Memory map and register definition . . . 1560
55.3.1 Memory map . . . 1560
55.3.2 Register descriptions . . . 1560

56 Mode Entry Module (MC_ME) . . . 1562
56.1 Introduction . . . 1562
56.1.1 Overview . . . 1562
56.1.2 Features . . . 1564
56.1.3 Modes of operation . . . 1564
56.2 External signal description . . . 1565
56.3 Memory map and register definition . . . 1565
56.3.1 Memory map . . . 1566
56.3.2 Register Description . . . 1569
56.4 Functional Description . . . 1594
56.4.1 Mode Transition Request . . . 1594
56.4.2 Modes Details . . . 1595
56.4.3 Mode Transition Process . . . 1598
56.4.4 Protection of Mode Configuration Registers . . . 1606
56.4.5 Mode Transition Interrupts . . . 1606
56.4.6 Peripheral Clock Gating . . . 1608
56.4.7 Application example . . . 1608

57 e200z215An3 Core Debug Support . . . 1610
57.1 Overview . . . 1610
57.1.1 Software debug facilities . . . 1610
57.1.2 Additional debug facilities . . . 1611
57.1.3 Hardware debug facilities . . . 1611
57.1.4 Sharing debug resources by software/hardware . . . 1612
57.2 Software debug events and exceptions . . . 1613
57.2.1 Instruction Address Compare event . . . 1614
57.2.2 Data Address Compare event . . . 1615
57.2.3 Linked Instruction Address Compare and Data Address Compare events . . . 1618

57.2.4 Trap debug event . . . 1618
57.2.5 Branch Taken debug event . . . 1618
57.2.6 Instruction Complete debug event . . . 1618
57.2.7 Interrupt Taken debug event . . . 1619
57.2.8 Critical Interrupt Taken debug event . . . 1619
57.2.9 Return debug event . . . 1619
57.2.10 Critical Return Debug Event . . . 1620
57.2.11 External debug event . . . 1620
57.2.12 Unconditional debug event . . . 1620
57.2.13 Performance Monitor Interrupt debug event . . . 1620
57.3 Debug registers . . . 1620
57.3.1 Debug Address and Value registers . . . 1621
57.3.2 Debug Control and Status registers . . . 1622
57.3.3 External Debug Resource Allocation Control (EDBRAC0) register . . . 1641
57.3.4 Debug Event Select register (DEVENT) register . . . 1650
57.3.5 Debug Data Acquisition Message (DDAM) register . . . 1650
57.4 Using debug resources for stack limit checking . . . 1651
57.5 External debug support . . . 1653
57.5.1 External debug registers . . . 1654
57.5.2 OnCE introduction . . . 1659
57.5.3 JTAG/OnCE pins . . . 1662
57.5.4 OnCE internal interface signals . . . 1662
57.5.5 OnCE interface signals . . . 1663
57.5.6 OnCE controller and serial interface . . . 1665
57.5.7 Access to debug resources . . . 1671
57.5.8 Methods of entering Debug mode . . . 1673
57.5.9 CPU Status and Control Scan Chain Register (CPUSCR) . . . 1674
57.5.10 Reserved registers (Reserved) . . . 1680
57.6 Watchpoint support . . . 1680
57.7 Basic steps for enabling, using, and exiting External Debug mode . . . 1683

58 Debug and Calibration Interface (DCI) . . . 1685
58.1 Introduction . . . 1685
58.1.1 Features . . . 1685
58.1.2 Overview . . . 1685

58.1.3 Operating modes 1687
58.2 External signal description 1693
58.3 Register description 1693
58.3.1 DCI control register (DCI_CR) 1693
58.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR) 1695

59 JTAG Controller (JTAGC) 1696
59.1 Introduction 1696
59.1.1 Block Diagram 1696
59.1.2 Features 1697
59.1.3 Modes of operation 1697
59.2 External signal description 1698
59.2.1 TCK—Test clock input 1698
59.2.2 TDI—Test data input 1698
59.2.3 TDO—Test data output 1698
59.2.4 TMS—Test mode select 1698
59.2.5 JCOMP—JTAG compliancy 1698
59.3 Register description 1699
59.3.1 Instruction register 1699
59.3.2 Bypass register 1699
59.3.3 Device identification register 1699
59.3.4 JTAG_PASSWORD register 1700
59.3.5 Boundary scan register 1701
59.4 Functional description 1701
59.4.1 JTAGC reset configuration 1701
59.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port 1701
59.4.3 TAP controller state machine 1701
59.4.4 JTAGC block instructions 1703
59.4.5 Boundary scan 1706
59.5 Initialization/Application information 1706

60 IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG) 1708
60.1 References 1708
60.2 Introduction 1708
60.2.1 Types of operation 1708

60.2.2 Deployment by class 1709
60.2.3 1149.7 TAP signals 1709
60.2.4 TAP.7 architecture 1710
60.2.5 Protocols 1711
60.2.6 Operating models 1712
60.3 CJTAG implementation summary 1712
60.3.1 T0 Functions 1712
60.3.2 T1 functions 1712
60.3.3 T2 functions 1712
60.3.4 T3 functions 1713
60.3.5 T4 functions 1713
60.4 Ancillary services 1713
60.4.1 Overview 1713
60.4.2 Resets 1713
60.4.3 Start-up Options 1717
60.4.4 RSU operation 1717
60.4.5 TAPC State Machine 1719
60.5 EPU (Extended Protocol Unit) Operation 1720
60.5.1 EPU Operation 1720
60.5.2 EPU Registers 1722
60.5.3 EPU Commands 1732
60.5.4 EPU operating states 1740
60.5.5 System and EPU Paths 1741
60.6 APU (Advanced Protocol Unit) Operation 1741
60.6.1 Overview 1741
60.6.2 Operation 1744
60.6.3 Escape sequences 1745
60.6.4 Signal behaviors 1745
60.6.5 APU Functions 1745
60.6.6 Configuration Change Packets (CP) 1752
60.6.7 Scan Packet 1752
60.6.8 SP Format 1752
60.7 Functional Description 1758
60.7.1 Switching from Standard Protocol to Advanced Protocol 1758

61 JTAG Data Communication (JDC) 1759

61.1 Introduction . . . 1759
61.2 Overview . . . 1759
61.3 Register definition . . . 1760
61.3.1 Register descriptions . . . 1760
61.3.2 Non-Memory mapped register definition . . . 1763
61.4 Functional description . . . 1764
61.4.1 IPS register access . . . 1764
61.4.2 JTAG register access . . . 1764

62 JTAG Master (JTAGM) . . . 1766
62.1 Introduction . . . 1766
62.1.1 Overview . . . 1766
62.1.2 Feature description . . . 1767
62.2 Functional description . . . 1767
62.2.1 Module architecture . . . 1767
62.2.2 JTAGM Ready signal . . . 1768
62.2.3 EVTO and EVTI signals . . . 1768
62.2.4 JTAGM configuration and status monitoring . . . 1769
62.2.5 JTAGM to DCI serial interface . . . 1769
62.2.6 Software interface . . . 1770
62.3 Modes of operation . . . 1770
62.4 Memory map and register definition . . . 1770
62.4.1 Register descriptions . . . 1770

63 Sequence Processing Unit (SPU) . . . 1778
63.1 Introduction . . . 1778
63.1.1 SPU block diagram . . . 1778
63.1.2 Overview . . . 1779
63.1.3 Features . . . 1780
63.2 SPU actions . . . 1781
63.3 Operation mode . . . 1782
63.3.1 SPU enable/disable . . . 1782
63.4 SPU module interface . . . 1782
63.5 Register definition . . . 1782
63.5.1 Register descriptions . . . 1787
63.6 Functional description . . . 1816

63.6.1 Input Mux unit . . . 1816
63.6.2 Processor exception vector encoding . . . 1818
63.6.3 State logic unit (SLU) . . . 1819
63.6.4 Sequence formation . . . 1820
63.6.5 Performance counters/timers unit . . . 1823
63.6.6 Action unit . . . 1824
63.6.7 Optional trace group . . . 1824
63.6.8 Format for CKSRC & CKDATA Values . . . 1825

64 Development Trigger Semaphore (DTS) . . . 1826
64.1 Introduction . . . 1826
64.2 Overview . . . 1826
64.3 DTS device connections . . . 1828
64.3.1 DTS register access . . . 1829
64.4 Memory map and register definition . . . 1829
64.4.1 Register descriptions . . . 1830
64.5 Example application . . . 1832

65 Nexus Aurora Router (NAR) . . . 1835
65.1 Introduction . . . 1835
65.2 Overview . . . 1836
65.3 Features . . . 1836
65.4 Register definition . . . 1836
65.4.1 Register descriptions . . . 1837
65.4.2 Message filtering registers . . . 1840
65.4.3 Trace memory bus configuration registers . . . 1843
65.4.4 Suppress mode triggers register (NAR_STCR) . . . 1845
65.4.5 Receive queue client disable register (NAR_CDR) . . . 1847
65.4.6 Trace Memory fill level and partition configuration register (NAR_AHFPAR) . . . 1849
65.5 Functional description . . . 1849
65.5.1 Reset/Startup . . . 1849
65.5.2 Operation modes . . . 1850
65.5.3 Data reception and arbitration . . . 1852
65.5.4 Suppress mode . . . 1853
65.5.5 Stall detection . . . 1854

65.5.6 Output arbitration . . . 1855
65.5.7 Message translation and formatting . . . 1856
65.5.8 Aurora interface . . . 1857
65.5.9 Trace memory bus output port usage . . . 1857
65.5.10 Internal message generation . . . 1858
65.5.11 Timestamp function . . . 1862
65.5.12 Overlay and internal trace memory full status generation . . . 1863
65.5.13 Configuration/debug interface JTAG (1149.1) . . . 1864
65.5.14 Trace memory bus controller . . . 1867
65.5.15 Legacy client interface controller . . . 1869
65.6 Initialization information . . . 1870
65.7 Application information . . . 1870

66 e200z215An3 Nexus 3 Module . . . 1871
66.1 Introduction . . . 1871
66.1.1 General Description . . . 1871
66.1.2 Terms and Definitions . . . 1871
66.1.3 Feature List . . . 1872
66.1.4 Functional Block Diagram . . . 1874
66.2 Enabling Nexus 3 Operation . . . 1874
66.2.1 Interaction with Low Power Modes . . . 1875
66.3 TCODEs supported . . . 1875
66.4 Nexus 3 Programmer's Model . . . 1881
66.4.1 Client Select Control (CSC) register . . . 1882
66.4.2 Port Configuration Register (PCR) — reference only . . . 1883
66.4.3 Nexus Development Control Register 1 (DC1) . . . 1884
66.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3) . . . 1885
66.4.5 Nexus Development Control Register 4 (DC4) . . . 1889
66.4.6 Development Status Register (DS) . . . 1890
66.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers . . . 1891
66.4.8 Nexus Watchpoint Mask (WMSK) register . . . 1896
66.4.9 Nexus Overrun Control (OVCR) register . . . 1897
66.4.10 Data Trace Control (DTC) register . . . 1898
66.4.11 Data Trace Start Address Registers (DTSA1–4) . . . 1900
66.4.12 Data Trace End Address 1–4 (DTEA1–4) registers . . . 1900
66.4.13 Read/Write Access Control/Status (RWCS) register . . . 1902

66.4.14 Read/Write Access Data (RWD) register 1903
66.4.15 Read/Write Access Address (RWA) 1905
66.5 Nexus 3 Register Access via JTAG/OnCE 1905
66.6 Nexus 3 Register Access via Software 1906
66.7 Nexus Message Fields 1906
66.7.1 TCODE Field 1906
66.7.2 Source ID Field (SRC) 1906
66.7.3 Relative Address Field (U-ADDR) 1906
66.7.4 Full Address Field (F-ADDR) 1907
66.8 Nexus Message Queues 1907
66.8.1 Message Queue Overrun 1908
66.8.2 CPU Stall 1908
66.8.3 Message Suppression 1908
66.8.4 Nexus Message Priority 1908
66.8.5 Data Acquisition Message Priority Loss Response 1910
66.8.6 Ownership Trace Message Priority Loss Response 1910
66.8.7 Program Trace Message Priority Loss Response 1910
66.8.8 Program Trace Sync Message Priority Loss Response 1910
66.8.9 Data Trace Message Priority Loss Response 1910
66.9 Debug Status messages 1910
66.10 Error messages 1911
66.11 Ownership trace 1911
66.11.1 Overview 1911
66.11.2 Ownership Trace Messaging (OTM) 1911
66.12 Program Trace 1912
66.12.1 Branch Trace Messaging types 1913
66.12.2 BTM Message For mats 1914
66.12.3 Program Trace Message Fields 1915
66.12.4 Resource Full Messages 1916
66.12.5 Program Correlation Messages 1917
66.12.6 Program Trace Overflow Error messages 1918
66.12.7 Program Trace Synchronization messages 1918
66.12.8 Program Trace Direct/Indirect Branch with Sync messages 1920
66.12.9 Enabling Program Trace 1921
66.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration) 1921

66.13 Data Trace 1922
66.13.1 Data Trace Messaging (DTM) 1923
66.13.2 DTM Message formats 1923
66.13.3 DTM operation 1925
66.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration) 1927
66.14 Data Acquisition messaging 1927
66.14.1 Data Acquisition ID Tag field 1927
66.14.2 Data Acquisition Data field 1927
66.14.3 Data Acquisition Trace event 1928
66.15 Watchpoint Trace messaging 1928
66.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration) 1929
66.16 External Hardware Trigger Controls 1930
66.17 Nexus 3 Read/Write access to memory-mapped resources 1931
66.17.1 Single write access 1931
66.17.2 Block write access 1932
66.17.3 Single read access 1933
66.17.4 Block read access 1933
66.17.5 Error handling 1934
66.17.6 Read/Write access error message 1934
66.18 Nexus 3 pin interface 1935
66.18.1 Pins implemented 1935
66.18.2 Pin protocol 1937
66.19 Rules for output messages 1939
66.20 Auxiliary port arbitration 1940
66.21 Examples 1940
66.22 Electrical characteristics 1943
66.23 IEEE 1149.1 (JTAG) RD/WR sequences 1943
66.23.1 JTAG sequence for accessing internal Nexus registers 1943
66.23.2 JTAG sequence for read access of memory-mapped resources 1944
66.23.3 JTAG sequence for write access of memory-mapped resources . . . 1944

67 Register Protection (REG_PROT) 1945

67.1 Introduction 1945
67.1.1 Overview 1945
67.1.2 Features 1945
67.1.3 Modes of operation 1946

67.2 External signal description . . . 1946
67.3 Memory map and register definition . . . 1946
67.3.1 Memory map . . . 1947
67.3.2 Register descriptions . . . 1948
67.4 Functional description . . . 1951
67.4.1 General . . . 1951
67.4.2 Change lock settings . . . 1951
67.4.3 Access errors . . . 1954
67.5 Initialization/application information . . . 1955
67.5.1 Reset . . . 1955

68 Password and Device Security Module (PASS) . . . 1956
68.1 Introduction . . . 1956
68.1.1 Overview . . . 1956
68.1.2 Features . . . 1956
68.1.3 Modes of operation . . . 1956
68.2 Memory map and register definition . . . 1957
68.2.1 Register descriptions . . . 1958
68.3 DCF clients . . . 1968
68.3.1 Censorship . . . 1968
68.4 Functional description . . . 1969
68.4.1 Censorship . . . 1969
68.4.2 Debug Interface Access . . . 1969
68.4.3 Flash Memory Read Protection . . . 1970
68.5 Initialization/application information . . . 1971
68.5.1 Reset . . . 1971
68.5.2 Modifying Life Cycle (maturing the device) . . . 1971
68.5.3 Setting lock bits in a password group . . . 1971
68.5.4 Censoring and uncensoring the device . . . 1972

69 Wakeup Unit (WKPU) . . . 1974
69.1 Introduction . . . 1974
69.1.1 Overview . . . 1974
69.1.2 Features . . . 1975
69.2 External signal description . . . 1975
69.3 Memory map and register description . . . 1975

69.3.1 Memory map 1975
69.3.2 Register descriptions 1976
69.4 Functional description 1982
69.4.1 Non-maskable interrupts 1982
69.4.2 External Wakeups/Interrupts 1982
69.4.3 Reset management 1983
69.4.4 System wakeup 1984
69.4.5 Glitch filter 1984
69.5 Initialization information 1984
69.5.1 Glitch filter and pad configuration 1984
69.5.2 Non-maskable interrupt initialization 1984
69.5.3 Reset request 1985

Appendix A Acronyms and abbreviations 1986

Revision history 1990

1 Preface

1.1 Overview

This document defines the functionality of the SPC572Lx microcontroller for use by software and hardware developers. The SPC572Lx is built on Power Architecture® technology and integrates technologies that are important for today's and tomorrow's automotive applications, including automotive powertrain controller applications for four-cylinder gasoline and diesel engines, chassis control applications, transmission control applications, steering and braking applications, as well as low-end hybrid applications and safety applications that 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*.

1.2 Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the SPC572Lx 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.

1.3 Document organization

This document includes chapters that are divided into parts:

- Part I includes chapters that describe the device as a whole or provide device-specific information.
- Part II includes chapters that describe the functionality of the individual modules on the device.

1.4 Register conventions

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

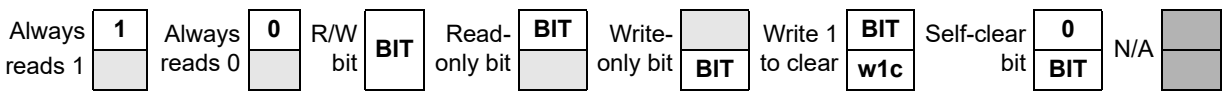


Figure 1. Key to register fields

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.

1.5 Acronyms and abbreviations

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

1.6 Reference documents

In addition to this reference manual, the following documents provide additional information on the operation of the SPC572Lx:

- 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 SPC572Lx microcontroller

The SPC572Lx microcontroller is a new member in a family of devices building on the legacy of the superseded SPC56xx family while introducing new features coupled with higher throughput to provide substantial reduction of cost per feature and significant power and performance improvement (MIPS per mW). There is one e200z215An3 processor core on the SPC572Lx device.

2.1.1 Target applications

This SPC57xx family of MCUs is targeted at automotive powertrain controller applications for gasoline and diesel engines, chassis control applications, transmission control applications, steering and braking applications, as well as low-end hybrid applications and two-wheelers.

The SPC572Lx device is specifically designed for 2-wheeler applications, predominantly motorcycles.

Many of these applications are considered to be functionally safe and are designed to be ISO26262 compliant. SPC572Lx implements the necessary architecture features to target ASIL-A integrity level.

2.1.2 Overall architecture

The shell, illustrated in *Figure 2*, incorporates the core, the platform and the I/O subsystem.

The architecture is based on an 80 MHz crossbar switch with master and slave ports. The master ports connected are the two CPU ports (32-bit load/store and 32-bit instruction) and a 32-bit 40 MHz concentrator port. A DMA unit, an LFAST interface and an Ethernet controller are connected on the concentrator. As slave ports are connected the flash memory controller, the SRAM controller and a Peripheral Bridge (PBRIDGE) to connect the peripheral cluster.

From a software perspective, every bus master (CPU, DMA, SIPI, and Ethernet) sees every memory and peripheral (all RAM modules, flash memory, and peripherals) in a consistent flat memory map.

2.1.3 Core features

The SPC572Lx CPU comprises a single 32-bit 80 MHz e200z215An3 core including a Nexus 3 module.

The CPU employs Power Architecture with enhancements that improve the architecture's fit in embedded applications. The Power Architecture VLE instruction set supports code size footprint reduction by encoding a mix of 16-bit and 32-bit instructions.

2.1.4 Memory hierarchy

The first level of memory is system RAM and flash memory. These are connected via an 80 MHz, 32-bit wide crossbar. The second level of memory system is the I/O subsystem. All the I/O peripherals are connected together via an 80 MHz 32-bit wide crossbar switch, while

peripherals are connected at either 80 MHz or 40 MHz depending on their need. The reduced speed for the peripherals helps to reduce power.

2.2 Features summary

On-chip modules within SPC572Lx include the following features:

- 1 main CPU, single issue, 32-bit CPU core complex (e200z2)
  - 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
  - Single-precision floating point operations
  - Saturation Instructions Extension adding scalar saturating arithmetic support to the *PowerISA* Integer Saturation (ISAT).
- 1568 KB (1536 KB code flash + 32 KB data flash) on-chip flash memory
  - Supporting multiple blocks allowing EEPROM emulation
  - RWW between data EEPROM and code flash memory
- 64 KB general-purpose data SRAM
- System Memory Protection Unit (SMPU)
- 16-channel Direct Memory Access controllers (eDMA) with two channel multiplexer for up to 60 DMA sources
- Interrupt Controller (INTC) supporting up to 1024 interrupt sources (all are not assigned)
- System Timer Module (STM)
- 2 Software Watchdog Timers (SWT)
- One Periodic Interrupt Timer (PIT) instance with four standard 32-bit timer channels
- One Periodic Interrupt Timer (PIT) instance with two 32-bit timer channels which can be combined into one 64-bit channel
- Single phase-locked loop with stable clock domain for peripherals and core (PLL)
- Single crossbar switch architecture for concurrent access to peripherals, flash memory, or SRAM from multiple bus masters
- System Integration Unit Lite (SIUL2)
- Boot Assist Flash (BAF) supports factory programming using a serial bootload through the UART Serial Boot Mode Protocol (physical interface (PHY) can be for example UART and CAN)
- PASS module (supporting 256-bit JTAG password protection)
- Device life cycle monitoring
- Generic Timer Module (GTM101)
- Enhanced analog-to-digital converter system with:
  - Three 12-bit SAR analog converters
  - One 16-bit Sigma-Delta analog converter
- Decimation unit to support SD ADC data conditioning
- 1 Deserial Serial Peripheral Interface (DSPI) module
- 2 LIN and UART communication interfaces (LINFlexD) modules
- 1 microsecond-bus channel (composed of one DSPI and one LINFlexD)

- 4 SENT (Single Edge Nibble Transmission) channels
- 2 Modular Controller Area Network (M_CAN) modules
- 1 Clock Calibration on CAN Unit (CCCU)
- Fast Ethernet Controller (FEC)
- Fast Asynchronous Serial Transmission (LFAST)
- Nexus Development Interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2010 standard
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1 and IEEE 1149.7)
- On-chip voltage regulator controller manages the supply voltage down to 1.2 V for core logic
- Self-test capability

Note: The source clock for this device can be: external oscillator, external crystal or 16 MHz internal RC oscillator (IRCOSC)

2.3 Feature list

Table 2 lists a summary of major features for the SPC572Lx 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. SPC572L64 device feature summary

Feature		Description
Process		55 nm
Main processor	Core	e200z2
	Number of main cores	1
	Single precision floating point	Yes
	VLE	Yes
Main processor frequency		80 MHz
SMPU		Yes
Software watchdog timer (task SWT/safety SWT)		2 (1/1)
Core Nexus class		3
Sequence processing unit (SPU)		Yes
System SRAM		64 KB
Flash memory		1536 KB
Flash memory fetch accelerator		8 × 128 bit
Data flash memory (EEPROM)		2 × 16 KB
Flash memory overlay RAM		8 KB
DMA channels		16
LINFlexD (UART/MSC)		3 (2/1)

Table 2. SPC572L64 device feature summary(Continued)

Feature	Description
M_CAN/M_TTCAN	2/0
DSPI (SPI/MSC/sync SCI)	2 (1/1/0)
Microsecond bus downlink	Yes
SENT bus	4
Ethernet	Yes
SIPI / LFAST Interprocessor bus	High speed (4-phase only)
System timers	4 PIT channels 1 AUTOSAR® (STM) 64-bit PIT
GTM timer	16 input channels, 56 output channels
GTM RAM	18.53 KB
Interrupt controller	1024 sources
ADC (SAR)	3
ADC (SD)	1
Temperature sensor	Yes
PLL	Single PLL with no FM
Internal linear voltage regulator	1.2 V
External power supplies	5 V 3.3 V(1)
Low-power modes	Stop mode Slow mode
Packages	eTQFP80 eTQFP100

1. Optional: can be used for special I/O segments.

2.4 Packaging

The SPC572Lx is available in these production packages: eTQFP80 and eTQFP100.

2.5 Software debug and calibration

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

- Nexus code execution trace on CPU and GTM
- Nexus data trace on CPU
- Sequence Processing Unit for complex trace triggering
- 8 KB overlay RAM
- debug over CAN

2.6 Block diagrams

Figure 2. Block diagram

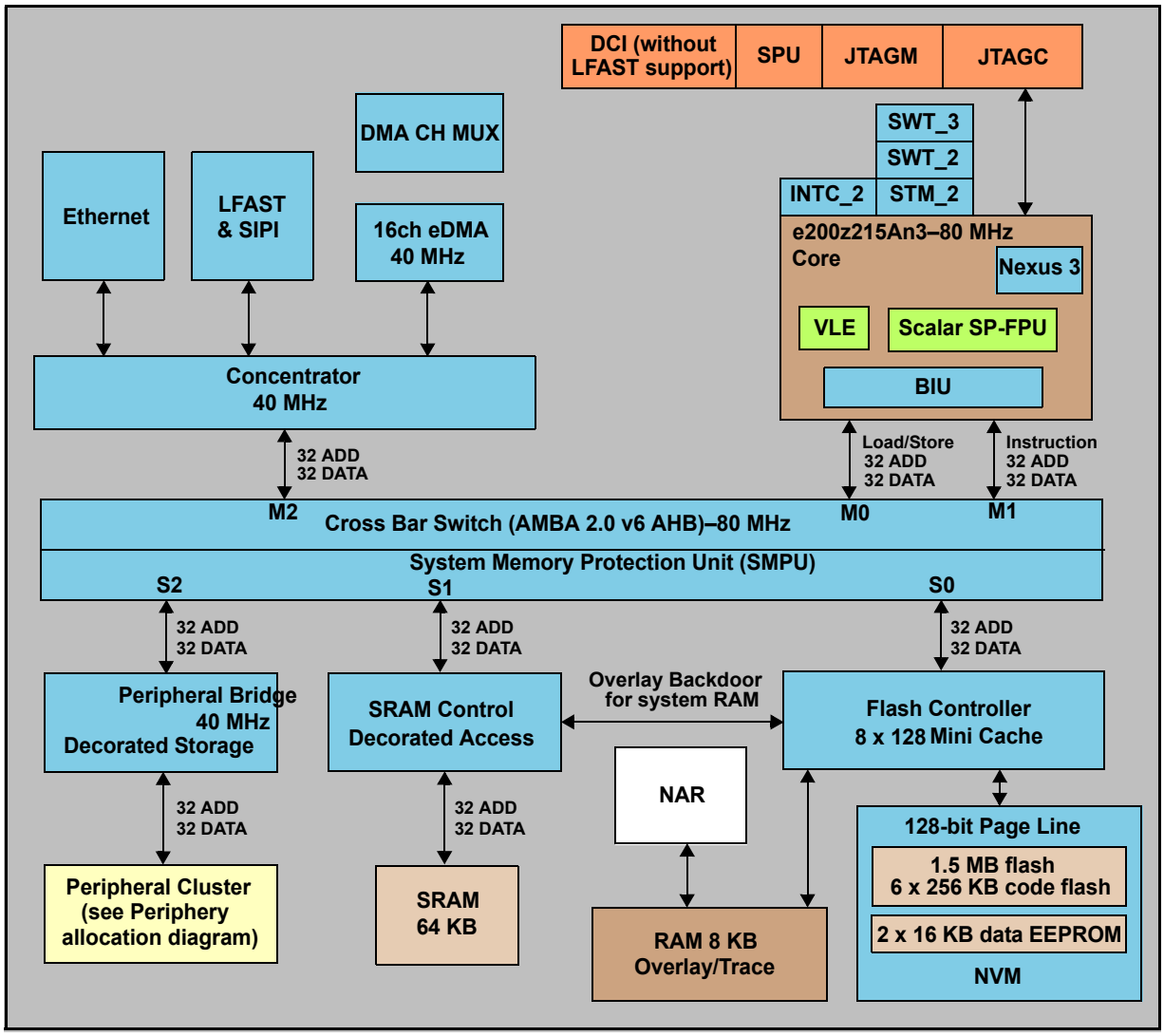
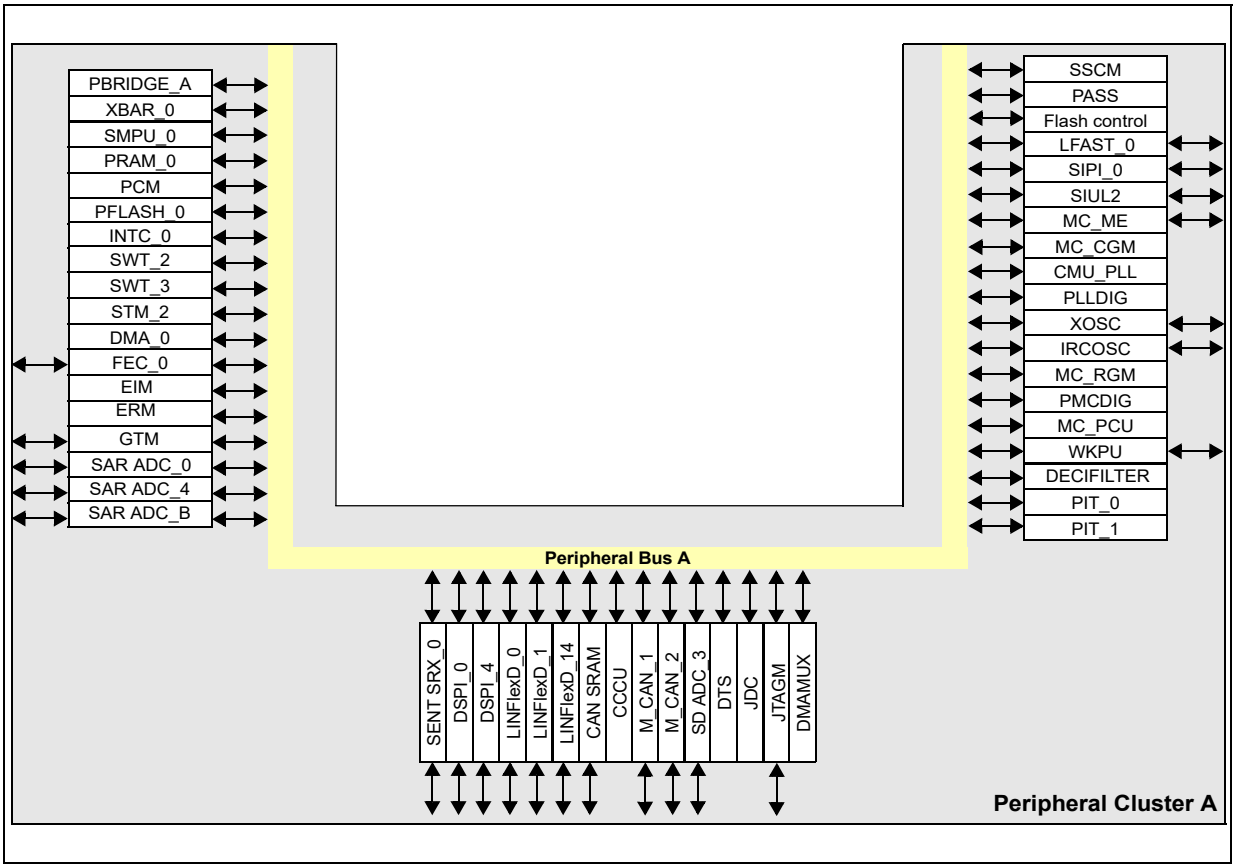


Figure 3. Periphery allocation



3 Embedded memories

3.1 Overview

The embedded memory architecture for SPC572Lx devices is a significant departure from previous-generation devices. Features include:

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

3.2 SRAM

3.2.1 System SRAM

SPC572Lx devices include 64 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 SRAM controller.

- See *Chapter 27: RAM controller (PRAM)* for details on configuring the SRAM controller.
- See *Table 10: RAM memory map* in *Chapter 5: Memory map* for the SPC572Lx SRAM map.

3.2.2 Overlay SRAM

Overlay SRAM is included in the SPC572Lx family of devices as part of a comprehensive set of calibration and debug features. Overlay SRAM can be mapped over specific regions of on-chip flash memory so that any access to an overlaid flash address is routed to the overlay SRAM instead. This enables the calibration of constant data without requiring additional external RAMs and calibration memory interfaces. The SPC572Lx has two 4 KB banks of ECC-protected RAM, for a total of 8 KB.

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

Overlay memory can be used to hold the device debug trace stream. Timing for calibration accesses to a particular overlay SRAM is the same as access to the underlying flash memory if the overlay SRAM is not being used as a destination for trace streaming at the same time.

a. Refer to the respective module chapter for details.

The calibration remap function supports two different types of overlay SRAM:

- An internal overlay RAM is included on all standard production devices.
- A portion of the system RAM can be used as the overlay RAM.

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

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

See *Section 28.5.12: PFlash calibration remap support*, for details on memory overlay (remapping).

3.2.3 Processor core local SRAM

SPC572Lx devices include local instruction and data SRAM (I-MEM and D-MEM) for each processor core to provide enhanced performance.

- Each main processor core has 16 KB of instruction SRAM and 64 KB of data SRAM.
- The peripheral processor core has 16 KB of instruction SRAM and KB data SRAM.

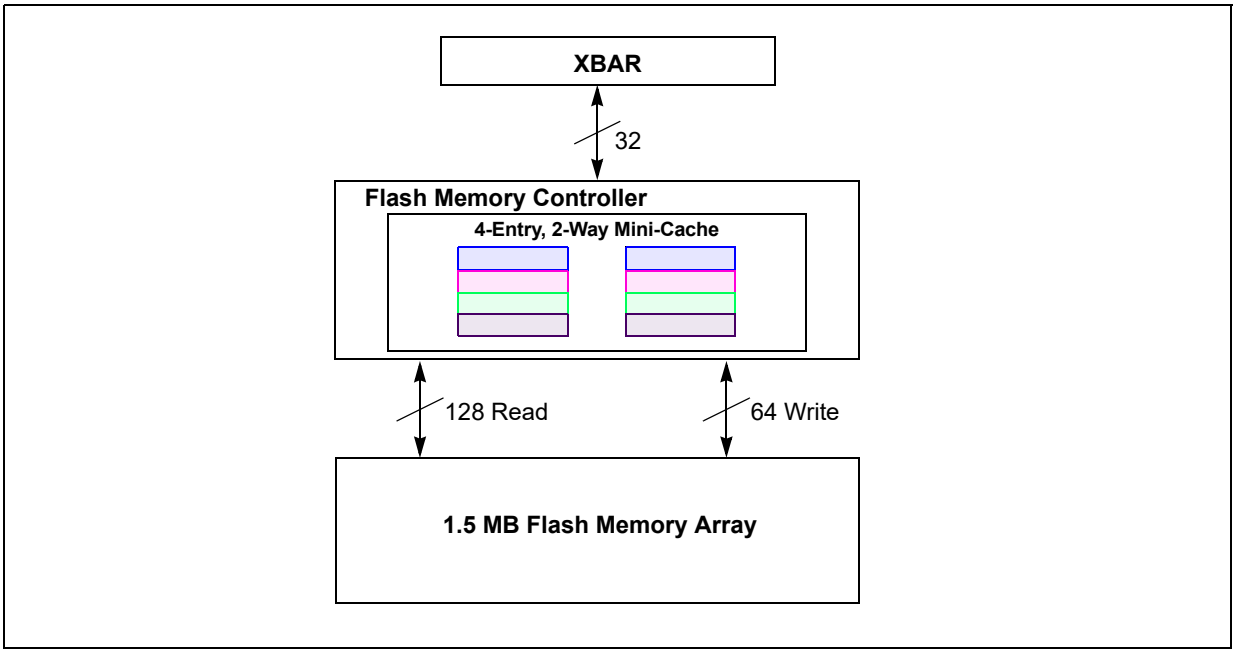
The core SRAM supports zero wait-state, single-cycle latency on processor local accesses.

Each area of core SRAM is accessible by other processor cores and bus mastering peripheral devices. See *Table 10: RAM memory map* in *Chapter 5: Memory map* for details.

3.3 Embedded flash memory

Flash memory on SPC572Lx 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 4* shows the memory architecture.

Figure 4. SPC572Lx 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 SPC572Lx flash memory controller acts as an interface between the system bus and the flash memory array and serves as the interface to the on-chip overlay RAM.

The flash memory controller contains a four-entry, two-way set-associative mini-cache that delivers flash read data with a zero-wait state response on lines that reside in the cache. Each entry contains one flash page, a 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, calibration RAM overlay remapping, and read wait state management of the flash memory.

See *Chapter 28: Flash memory controller (PFLASH Controller)*, for details.

3.3.2 Flash memory array

SPC572Lx devices include a single 1.5 MB flash memory array, referred to as “main space,” plus an independent 8 KB block of one-time-programmable (OTP) flash memory included to support systems that require non-volatile memory for security features or system initialization information. The independent 8 KB block is referred to as “UTEST space.”

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
  - 256 KB blocks provided for code or data
  - Two 16 KB blocks provided for EEPROM emulation
  - 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 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 three groupings:

- 16 KB low address space
- 16 KB high address space
- 256 KB address space

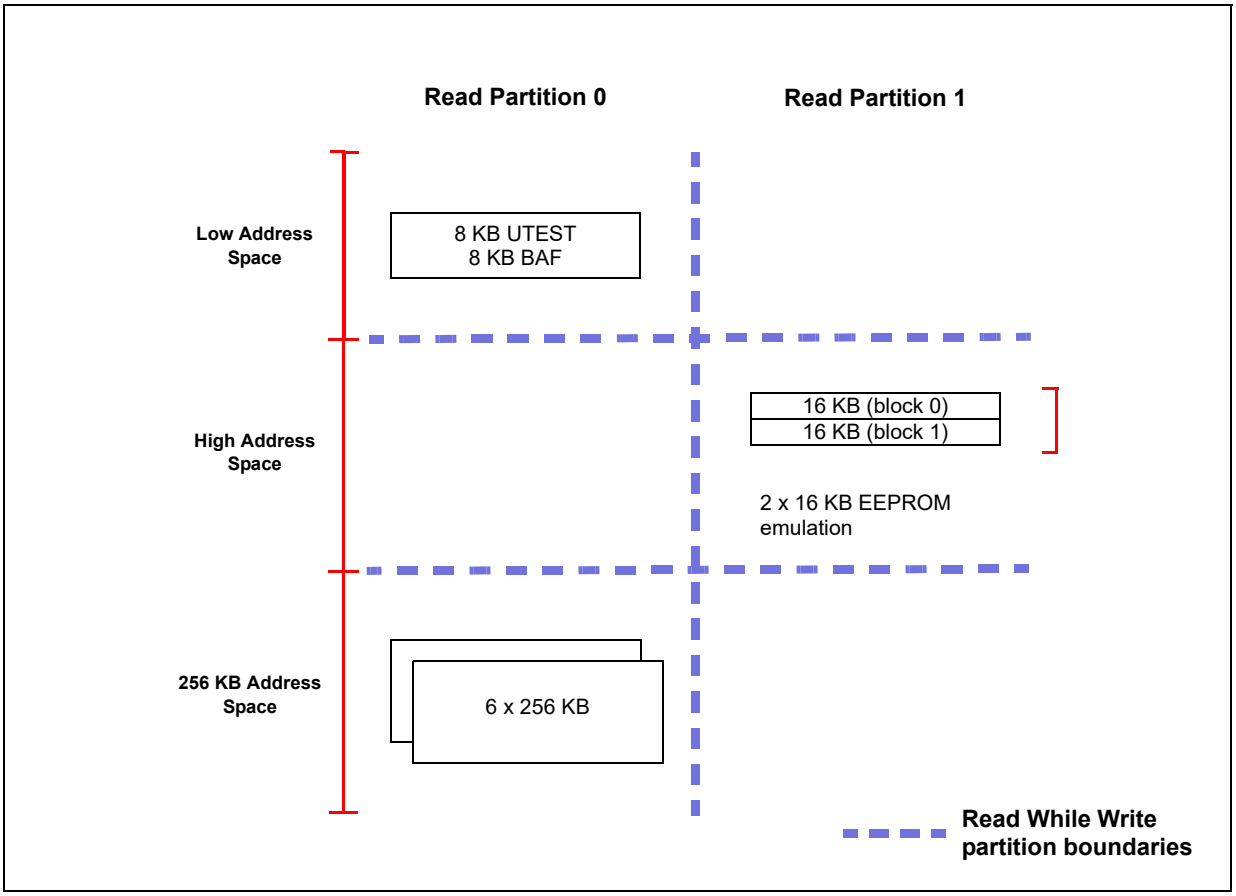
Address space group (low- or high-address space) determines which registers/fields are used to select blocks for modify operations or lock them from modify operations. Each address space is independent.

The flash module is further divided into 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 5 shows the flash block segmentation and read-while-write partitioning for SPC572Lx devices.

Figure 5. Flash memory segmentation and read-while-write partitioning



Note: See Table 9: Flash memory and overlay RAM map in Chapter 5: Memory map for the SPC572Lx flash memory map.

3.3.2.3 UTEST memory space

Devices in the SPC572Lx family contain an 8 KB area of One-Time Programmable (OTP) flash memory for storing test information and device configuration data. See *Table 11: UTEST flash memory map* in *Chapter 5: Memory map* for the SPC572Lx UTEST flash memory map.

3.4 Security features

The SPC572Lx 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 Production packages

The following production packages are for the SPC572Lx device:

- eTQFP80
- eTQFP100

Case number and outline drawings for each package are provided in the SPC572Lx data sheet.

4.2 Package pinouts

For package pinouts refer to the SPC572Lx data sheet.

4.3 Pin descriptions

The following sections provide signal descriptions and related information about the functionality and configuration of the SPC572Lx device. The signal multiplexing options appear in the “I/O signal description and input multiplexing” table[b].

4.3.1 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
ESR0	Output	Output, open-drain	Input, weak pullup
ESR1	Weak pulldown	Input, weak pulldown	Input, weak pulldown
PORST	Output pulldown	Input, weak pulldown	Input, weak pulldown
TESTMODE(4)	Weak pulldown	Input, weak pulldown	Input, weak pulldown
JCOMP (TRST)	High-impedance	Input, weak pulldown	Input, weak pulldown
TDI	High-impedance	Input, weak pullup	Input, weak pullup
TDO	High-impedance	Weak pullup	High-impedance
TMS	High-impedance	Input, weak pulldown	Input, weak pulldown
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.

b. The “I/O signal description and input multiplexing” table is located in a Microsoft Excel® workbook attached to this document.

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.

4.3.2 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. The Supply Pins Table contains information on power supply and reference pins. See the Signal Table (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the excel file to open it and select the Supply Pins Table tab.

Table 4. Power supply and reference pins

Supply			QFP pin	
Symbol	**Type**	**Description**	**80**	**100**
V_{SS_HV}	Ground	High voltage ground	Exposed pad 81	Exposed pad 101
V_{SS_LV}	Ground	Low voltage ground	Exposed pad 81	Exposed pad 101
$V_{SS_HV_OSC}$	Ground	Ground supply for the oscillator	Exposed pad 81	Exposed pad 101
V_{DD_LV}	Power	Low voltage power supply for production device (PLL is also powered by this pin.)	15, 42, 68	19, 52, 68
$V_{DD_HV_PMC}$	Power	High voltage power supply for internal power management unit	55	—
$V_{DD_HV_IO_MAIN}$	Power	High voltage power supply for I/O	16, 41, 67	20, 51, 85
$V_{DD_HV_IO_JTAG}$	Power	JTAG/Oscillator power supply	45	55
$V_{DD_HV_OSC}$	Power	Oscillator voltage supply	45	55
$V_{DD_HV_IO_ETH}$	Power	Ethernet 3.3 V I/O supply	77	95
$V_{DD_HV_FLA}$	Power	Decoupling supply pin for flash	68	86
$V_{DD_HV_ADV}$	Power	High voltage supply for ADC	33	39
$V_{SS_HV_ADR}$	Reference	Ground reference of ADCs	23	28
$V_{DD_HV_ADR}$	Reference	Voltage reference of ADCs	24	29

4.3.3 System pins

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

Table 5. System pins

Symbol	Description	Direction	QFP pin	
			80	100
PORST	Power on reset with Schmitt trigger characteristics and noise filter. PORST is active low	Bidirectional	56	66
ESR0	External functional reset with Schmitt trigger characteristics and noise filter. ESR0 is active low	Bidirectional	57	67
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	54	64
XTAL	Analog output of the oscillator amplifier circuit needs to be grounded if oscillator is used in bypass mode.	Output	44	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	43	53

4.3.4 LVDS pins

Table 6 contains information on LVDS pin functions for the devices.

Table 6. LVDSM pin descriptions

Functional block	Port pin	Signal	Signal description	Direction	Package pin number	
					eTQFP80	eTQFP100
SIPI LFAST(1)	PF[13]	SIPI_RXN	Interprocessor Bus LFAST, LVDS Receive Negative Terminal	I	46	56
	PD[7]	SIPI_RXP	Interprocessor Bus LFAST, LVDS Receive Positive Terminal	I	47	57
	PD[6]	SIPI_TXN	Interprocessor Bus LFAST, LVDS Transmit Negative Terminal	O	48	58
	PA[8]	SIPI_TXP	Interprocessor Bus LFAST, LVDS Transmit Positive Terminal	O	49	59

Table 6. LVDSM pin descriptions(Continued)

Functional block	Port pin	Signal	Signal description	Direction	Package pin number	
					eTQFP80	eTQFP100
DSPI 4 Microsecond Bus	PD[3]	SCK_N	DSPI 4 Microsecond Bus Serial Clock, LVDS Negative Terminal	O	69	87
	PD[2]	SCK_P	DSPI 4 Microsecond Bus Serial Clock, LVDS Positive Terminal	O	70	88
	PD[1]	SOUT_N	DSPI 4 Microsecond Bus Serial Data, LVDS Negative Terminal	O	71	89
	PD[0]	SOUT_P	DSPI 4 Microsecond Bus Serial Data, LVDS Positive Terminal	O	72	90

1. DRCLK and TCK/DRCLK usage for SIPI LFAST is described in the SPC572Lx reference manual, refer to SIPI LFAST chapter.

4.3.5 Generic pins

The Signal Description table contains the port definition, multiplexing, direction, pad type, and package pin numbers for each I/O pin on the device. Multi-function pins are programmable via their respective Multiplexed Signal Configuration Register (MSCR) values.

Note: The SPC572Lx "I/O signal description and input multiplexing table is contained in a Microsoft Excel® workbook file attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the excel file to open it and select the I/O Signal Description Table tab.

- *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).*

5 Memory map

Table 7 shows the device memory map for the SPC572Lx.

All addresses on the SPC572Lx, including those that are reserved, are identified in the table. The addresses represent the physical addresses assigned to each region or module name. All memory not listed in this table is reserved, and access to those locations may produce undesirable results.

Table 7. Overview memory map

Start address	End address	Allocated size	Used size	Description
Flash memory (XBAR port 0 – 0)				
0x00000000	0x003FFFFF	—	Reserved	
0x00400000	0x005FFFFF	2 MB	8 KB	UTEST NVM Block – no overlay
0x00600000	0x007FFFFF	—	Reserved	
0x00800000	0x00807FFF	32 KB	32 KB	Data Flash Blocks – no overlay
0x00808000	0x009FFFFF	—	Reserved	
0x01000000	0x01FFFFFF	16 MB	1536 KB	Large Flash Blocks – no overlay
0x02000000	0x08FFFFFF	—	Reserved	
0x09000000	0x09FFFFFF	16 MB	1536 KB	Large Flash Blocks – overlay enabled mapping
0x0A000000	0x0CFFFFFF	—	Reserved	
0x0D000000	0x0D003FFF	16 KB	8 KB	Internal Overlay RAM
0x0D004000	0x3FFFFFFF	—	Reserved	
System RAM (XBAR Slave Port 0 – 4)				
0x40000000	0x401FFFFF	2048 KB	64 KB	System RAM
0x40200000	0xEFFFFFFF	—	Reserved	
Peripherals PBRIDGE_A, PBRIDGE_B (XBAR Slave Port 1 – F)				
0xF0000000	0xFFFFFFFF	256 MB	—	Peripheral address space(1)

1. See *Table 8* for details.

Table 8. Peripheral (PBRIDGE_A) memory map

Start address	End address	Allocated size	Used size	PBRIDGE Access Control Register	Description
0xF0000000	0xFBFFFFFF	—	Reserved		
0xFC000000	0xFC003FFF	16 KB	—	PACR0	Peripheral Bridge A (PBRIDGE_A)
0xFC004000	0xFC007FFF	16 KB	—	PACR1	Crossbar (XBAR_0)

Table 8. Peripheral (PBRIDGE_A) memory map(Continued)

Start address	End address	Allocated size	Used size	PBRIDGE Access Control Register	Description
0xFC008000	0xFC00FFFF	—	Reserved		
0xFC010000	0xFC013FFF	16 KB	—	PACR4	System Memory Protection Unit 0 (SMPU_0)
0xFC014000	0xFC01FFFF	—	Reserved		
0xFC020000	0xFC023FFF	16 KB	—	PACR8	PRAM (Platform RAM controller)
0xFC024000	0xFC027FFF	—	Reserved		
0xFC028000	0xFC02BFFF	16 KB	—	PACR10	Platform Configuration Module (PCM)
0xFC02C000	0xFC02FFFF	—	Reserved		
0xFC030000	0xFC033FFF	16 KB	—	PACR12	PFLASH (Platform Flash controller)
0xFC034000	0xFC03FFFF	—	Reserved		
0xFC040000	0xFC043FFF	16 KB	—	PACR16	Interrupt Controller (INTC_0)
0xFC044000	0xFC057FFF	—	Reserved		
0xFC058000	0xFC05BFFF	16 KB	—	PACR22	Software Watchdog Timer 2 (SWT_2)
0xFC05C000	0xFC05FFFF	16 KB	—	PACR23	Software Watchdog Timer 3 (SWT_3)
0xFC060000	0xFC06FFFF	—	Reserved		
0xFC070000	0xFC073FFF	16 KB	—	PACR28	System Timer Module 2 (STM_2)
0xFC074000	0xFC09FFFF	—	Reserved		
0xFC0A0000	0xFC0A3FFF	16 KB	—	PACR40	Enhanced Direct Memory Access 0 (eDMA_0)
0xFC0A4000	0xFC0AFFFF	—	Reserved		
0xFC0B0000	0xFC0B3FFF	16 KB	—	PACR44	Fast Ethernet Controller (FEC)
0xFC0B4000	0xFC0DBFFF	—	Reserved		
0xFC0DC000	0xFC0DFFFF	16 KB	—	PACR55	Error Injection Module (EIM)
0xFC0E0000	0xFC0E3FFF	16 KB	—	PACR56	Error Reporting Module (ERM)
0xFC0E4000	0xFFCFFFFF	—	Reserved		
0xFFD00000	0xFFDBFFFF	768 KB	—	PCTL144–PCTL191	Generic Timer Unit 101 Integration Module (GTMINT)
0xFFDC0000	0xFFDFFFFF	—	Reserved		
0xFFE00000	0xFFE03FFF	16 KB	—	PCTL127	SAR ADC 0 (SARADC_0)
0xFFE04000	0xFFE0FFFF	—	Reserved		
0xFFE10000	0xFFE13FFF	16 KB	—	PCTL123	SAR ADC 4 (SARADC_4)
0xFFE14000	0xFFE3BFFF	—	Reserved		
0xFFE3C000	0xFFE3FFFF	16 KB	—	PCTL112	SARADC_B
0xFFE40000	0xFFE5BFFF	—	Reserved		

Table 8. Peripheral (PBRIDGE_A) memory map(Continued)

Start address	End address	Allocated size	Used size	PBRIDGE Access Control Register	Description
0xFFE5C000	0xFFE5FFFF	16 KB	—	PCTL104	SENT (SAE J2716) Receiver 0 (SRX_0)
0xFFE60000	0xFFE6FFFF	—	Reserved		
0xFFE70000	0xFFE73FFF	16 KB	—	PCTL99	Deserial Serial Peripheral Interface 0 (DSPI_0)
0xFFE74000	0xFFE77FFF	—	Reserved		
0xFFE78000	0xFFE7BFFF	16 KB	—	PCTL97	Deserial Serial Peripheral Interface 4 (DSPI_4)
0xFFE7C000	0xFFE8BFFF	—	Reserved		
0xFFE8C000	0xFFE8FFFF	16 KB	—	PCTL92	LIN Controller 0 (LINFlexD_0)
0xFFE90000	0xFFE93FFF	16 KB	—	PCTL91	LIN Controller 1 (LINFlexD_1)
0xFFE94000	0xFFEA7FFF	—	Reserved		
0xFFEA8000	0xFFEABFFF	16 KB	—	PCTL85	LIN Controller 14 (LINFlexD_14)
0xFFEAC000	0xFFED3FFF	—	Reserved		
0xFFED4000	0xFFED7FFF	16 KB	3.16 KB	PCTL74	Shared CAN Message RAM
0xFFED8000	0xFFEE3FFF	—	Reserved		
0xFFEE4000	0xFFEE7FFF	16 KB	—	PCTL70	CAN Subsystem: Controller Area Network 1 (M_CAN_1)
0xFFEE8000	0xFFEEBFFF	16 KB	—	PCTL69	CAN Subsystem: Controller Area Network 2 (M_CAN_2)
0xFFEEC000	0xFFF03FFF	—	Reserved		
0xFFF04000	0xFFF07FFF	16 KB	—	PCTL62	Calibration on CAN Unit (CCCU)
0xFFF08000	0xFFF0FFFF	—	Reserved		
0xFFF10000	0xFFF13FFF	16 KB	—	PCTL59	Sigma-Delta (SD) SD ADC3
0xFFF14000	0xFFF37FFF	—	Reserved		
0xFFF38000	0xFFF3BFFF	16 KB	—	PCTL49	Development Tools Semaphore (DTS)
0xFFF3C000	0xFFF3FFFF	16 KB	—	PCTL48	JTAG Data Communication (JDC) Module
0xFFF40000	0xFFF47FFF	—	Reserved		
0xFFF48000	0xFFF4BFFF	16 KB	—	PCTL45	JTAG Master (JTAGM)
0xFFF4C000	0xFFF6BFFF	—	Reserved		
0xFFF6C000	0xFFF6C1FF	512 bytes	—	PCTL36	Direct Memory Access Controller Hardware Multiplexer (DMAMUX_0)
0xFFF6C200	0xFFF6C3FF	512 bytes	—	PCTL36	Direct Memory Access Controller Hardware Multiplexer (DMAMUX_1)
0xFFF6C400	0xFFF7FFFF	—	Reserved		

Table 8. Peripheral (PBRIDGE_A) memory map(Continued)

Start address	End address	Allocated size	Used size	PBRIDGE Access Control Register	Description
0xFFF80000	0xFFF83FFF	16 KB	—	PCTL31	Periodic Interrupt Timer 1 (PIT_1)
0xFFF84000	0xFFF87FFF	16 KB	—	PCTL30	Periodic Interrupt Timer 0 (PIT_0)
0xFFF88000	0xFFF8BFFF	16 KB	—	PCTL29	Decimation Filter (DECFILTER)
0xFFF8C000	0xFFF97FFF	—	Reserved		
0xFFF98000	0xFFF9BFFF	16 KB	—	PCTL25	Wakeup Unit (WKPU)
0xFFF9C000	0xFFF9FFFF	—	Reserved		
0xFFFA0000	0xFFFA03FF	1 KB	—	PCTL23	Power Control Unit (MC_PCU)
0xFFFA0400	0xFFFA3FFF	15 KB	—	PCTL23	Power Management Controller (PMCDIG)
0xFFFA4000	0xFFFA7FFF	—	Reserved		
0xFFFA8000	0xFFFABFFF	16 KB	—	PCTL21	Reset Generation Module (MC_RGM)
0xFFFAC000	0xFFFAFFFF	—	Reserved		
0xFFFB0000	0xFFFB003F	64 bytes	—	PCTL19	IRCOSC (Internal RC Oscillator)
0xFFFB0040	0xFFFB007F	—	Reserved		
0xFFFB0080	0xFFFB00BF	64 bytes	—	PCTL19	XOSC (Externally driven oscillator)
0xFFFB00C0	0xFFFB00FF	—	Reserved		
0xFFFB0100	0xFFFB013F	64 bytes	—	PCTL19	Phase-locked Loop Module (PLL)
0xFFFB0140	0xFFFB01FF	—	Reserved		
0xFFFB0200	0xFFFB023F	64 bytes	—	PCTL19	Clock Monitor Unit for PLL (CMU_PLL)
0xFFFB0240	0xFFFB06FF	—	Reserved		
0xFFFB0700	0xFFFB3FFF	14 KB	—	PCTL19	Clock Generation Module (MC_CGM)
0xFFFB4000	0xFFFB7FFF	—	Reserved		
0xFFFB8000	0xFFFBBFFF	16 KB	—	PCTL17	Mode Entry Module (MC_ME)
0xFFFBC000	0xFFFBFFFF	—	Reserved		
0xFFFC0000	0xFFFC3FFF	16 KB	—	PCTL15	System Integration Unit Lite 2 (SIUL2)
0xFFFC4000	0xFFFCFFFF	—	Reserved		
0xFFFD0000	0xFFFD3FFF	16 KB	—	PCTL11	Serial Interprocessor Interface 0 (SIPI_0)
0xFFFD4000	0xFFFD7FFF	—	Reserved		
0xFFFD8000	0xFFFDBFFF	16 KB	—	PCTL9	LFAST 0 (LFAST_0) (Interprocessor communication)
0xFFFDC000	0xFFFDFFFF	—	Reserved		
0xFFFE0000	0xFFFE3FFF	16 KB	—	PCTL7	Flash Main Control Registers
0xFFFE4000	0xFFFF3FFF	—	Reserved		

Table 8. Peripheral (PBRIDGE_A) memory map(Continued)

Start address	End address	Allocated size	Used size	PBRIDGE Access Control Register	Description
0xFFFF4000	0xFFFF7FFF	16 KB	—	PCTL2	Password and Device Security Module (PASS)
0xFFFF8000	0xFFFFBFFF	16 KB	—	PCTL1	System Status and Configuration Module (SSCM)
0xFFFFC000	0xFFFFFFFF	16 KB	—	PCTL0	Boot Assist ROM (BAR)

Table 9 contains the detailed flash memory map for the SPC572Lx.

Table 9. Flash memory and overlay RAM map

Start address	End address	Allocated size	Complete flash memory block structure	RWW partition	Block size
Reserved – no overlay					
0x00000000	0x003FFFFF	—	Reserved		
UTEST NVM Block – no overlay					
0x00400000	0x00403FFF	16 KB	UTEST NVM Block Space 16 KB	0	8 KB
0x00404000	0x00407FFF	16 KB	BAF (block0)	0	8 KB
0x00408000	0x007FFFFF	—	Reserved		
Data Flash– no overlay					
0x00800000	0x00803FFF	16 KB	EEPROM block0	1	16 KB
0x00804000	0x00807FFF	16 KB	EEPROM block1	1	16 KB
0x00808000	0x00FFFFFF	—	Reserved		
Large flash memory blocks – no overlay					
0x01000000	0x0103FFFF	256 KB	256 KB Flash block0	0	256 KB
0x01040000	0x0107FFFF	256 KB	256 KB Flash block1	0	256 KB
0x01080000	0x010BFFFF	256 KB	256 KB Flash block2	0	256 KB
0x010C0000	0x010FFFFF	256 KB	256 KB Flash block3	0	256 KB
0x01100000	0x0113FFFF	256 KB	256 KB Flash block4	0	256 KB
0x01140000	0x0117FFFF	256 KB	256 KB Flash block5	0	256 KB
0x01180000	0x08FFFFFF	—	Reserved		
Large flash blocks – overlay enabled mapping					
0x09000000	0x0903FFFF	256 KB	256 KB Flash block0	0	256 KB
0x09040000	0x0907FFFF	256 KB	256 KB Flash block1	0	256 KB
0x09080000	0x090BFFFF	256 KB	256 KB Flash block2	0	256 KB

Table 9. Flash memory and overlay RAM map(Continued)

Start address	End address	Allocated size	Complete flash memory block structure	RWW partition	Block size
0x090C0000	0x090FFFFF	256 KB	256 KB Flash block3	0	256 KB
0x09100000	0x0913FFFF	256 KB	256 KB Flash block4	0	256 KB
0x09140000	0x0917FFFF	256 KB	256 KB Flash block5	0	256 KB
0x09180000	0x0CFFFFFF	—	Reserved		
Internal Overlay RAM					
0x0D000000	0x0D003FFF	16 KB	Internal Overlay RAM	—	8 KB
0x0D004000	0x0FFFFFFF	—	Reserved		

Table 10 details the system RAM memory map.

Table 10. RAM memory map

Start address	End address	Allocated size	Used	Description
System RAM (XBAR Port 0 – 4)				
0x40000000	0x401FFFFF	2048 KB	64 KB	System RAM
0x40200000	0x5FFFFFFF	—	Reserved	

Table 11. UTEST flash memory map

Start address	End address	Allocated size [bytes]	Description	Comments
UTEST flash memory				
0x00400000	0x00400001	2	Temperature Sensor Calibration A	Programmed by during factory test. Contains the temperature sensor and bandgap ADC readings to be used by software to improve the accuracy of the temp sensor readings
0x00400002	0x00400003	2	Temperature Sensor Calibration B	Programmed during factory test. Contains the temperature sensor and bandgap ADC readings to be used by software to improve the accuracy of the temp sensor readings
0x00400004	0x0040000B	—	Reserved	
0x0040000C	0x0040000F	4	Test Mode Disable Seal	Programmed by the customer to disable factory test mode access

Table 11. UTEST flash memory map(Continued)

Start address	End address	Allocated size [bytes]	Description	Comments
0x00400010	0x0040001F	16	Test Mode Disable Block Select Group A	Programmed by the customer to select which blocks cannot be tested in Failure Analysis
0x00400020	0x0040002F	16	Factory Erase diary Location	Programmed by the customer before one-time factory erase is allowed
0x00400030	0x0040003F	16	Test Mode Disable Block Select Group B	Secondary location that can be programmed by the customer to select which blocks cannot be tested in Failure Analysis
0x00400040	0x0040005F	32	Customer Single Bit Correction Area	Programmed by factory to include ECC/EDC errors to allow testing of ECC/EDC hardware
0x00400060	0x0040007F	32	Customer Double Bit Detection Area	—
0x00400080	0x0040009F	32	Customer EDC after ECC Area	—
0x004000A0	0x004000BF	32	Unique Identifier (UID)	Programmed during factory test. The UID includes information on Wafer lot, X/Y-position on the wafer, manufacturing data, test results, flash firmware ID and parameters, company ID: 66 = STMicroelectronics
0x004000C0	0x004000FF	—	Reserved	
0x00400100	0x00400103	4	Test Mode Override Passcode	Protected from read operations by flash BIU from Production at OEM Life Cycle. Programmed by customer. Becomes unreadable (hidden) as soon as customer programs lifecycle stage to ‘Production at OEM’ or later
0x00400104	0x0040011F	—	Reserved	
0x00400120	0x0040013F	32	JTAG Password	Protected from read operations by flash BIU from Production at OEM Life Cycle. Programmed by customer. Becomes unreadable (hidden) as soon as customer programs lifecycle stage to ‘Production at OEM’ or later
0x00400140	0x0040015F	32	PASS Password Group 0	
0x00400160	0x0040017F	32	PASS Password Group 1	
0x00400180	0x0040019F	32	PASS Password Group 2	
0x004001A0	0x004001BF	32	PASS Password Group 3	
0x004001C0	0x004001FF	—	Reserved	

Table 11. UTEST flash memory map(Continued)

Start address	End address	Allocated size [bytes]	Description	Comments
0x00400200	0x0040020F	16	Lifecycle Slot 0 - JDP_PROD	JDP Production Lifecycle slot - Programmed by JDP during test
0x00400210	0x0040021F	16	Lifecycle Slot 0 - CUST_DEL	Customer Delivery Lifecycle slot - Programmed by JDP during test
0x00400220	0x0040022F	16	Lifecycle Slot 0 - OEM_PROD	OEM Production Lifecycle slot - Programmed by customer during ECO module production/test
0x00400230	0x0040023F	16	Lifecycle Slot 0 - IN_FIELD	In Field Lifecycle slot - Programmed by customer during ECO module production/test
0x00400240	0x0040024F	16	Lifecycle Slot 0 - FA	FA Lifecycle slot - Programmed by customer on returned modules
0x00400250	0x0040025F	—	Reserved	
0x00400300	0x00400FFF	3328	DCF Start Record	DCF Start Record. Programmed by JDP during test. Contiguous list of DCF records start at this address.
0x00401000	0x00401FFF	4096	Reserved for customer OTP data	Programmed by the customer

6 Device configuration

6.1 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
- links to further information.

6.2 Core module

SPC572Lx has one core (e200z215An3) that performs various computational and control functions. The core has no local cache.

6.2.1 Core reset settings

Table 12 shows the state of the *PowerISA 2.06* architected registers and other optional resources immediately following a system reset.

Table 12. Reset settings for e200z215An3 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)
DAC3	0x0000_0000(3)
DAC4	0x0000_0000(3)
DBCR0	0x0000_0000(3)
DBCR1	0x0000_0000(3)
DBCR2	0x0000_0000(3)
DBCR4	0x0000_0000(3)
DBCR5	0x0000_0000(3)
DBCR6	0x0000_0000(3)
DBCR7	0x0000_0000(3)
DBCR8	0x0000_0000(3)

Table 12. Reset settings for e200z215An3 resources(Continued)

Resource	System reset setting
DBSR	0x1000_0000(3)
DDAM	0x0000_0000(3)
DDEAR	Unaffected(2)
DEAR	Unaffected(2)
DEVENT	0x0000_0000(3)
DSRR0	Unaffected(2)
DSRR1	Unaffected(2)
DVC1	Unaffected(2)
DVC2	Unaffected(2)
ESR	0x0000_0000
HID0	0x0000_0000
HID1	0x0000_0000
IAC1	0x0000_0000(3)
IAC2	0x0000_0000(3)
IAC3	0x0000_0000(3)
IAC4	0x0000_0000(3)
IAC5	0x0000_0000(3)
IAC6	0x0000_0000(3)
IAC7	0x0000_0000(3)
IAC8	0x0000_0000(3)
IVPR	Unaffected(2)
LR	Unaffected(2)
MAS1	Unaffected(2)
MAS2	Unaffected(2)
MAS3	Unaffected(2)
MAS4	Unaffected(2)
MCAR	Unaffected(2)
MCSR	0x0000_0000
MCSRR0	Unaffected(2)
MCSRR1	Unaffected(2)
MMUCFG(4)	0x0000_0003
MPU0CSR0	0x0000_0000
MPU0CFG(4)	—
MSR	0x0000_0000
NPIDR	0x0000_0000

Table 12. Reset settings for e200z215An3 resources(Continued)

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

1. The levels that determine these bits are set by the factory and may change between revisions of the device.
   Core: p_cpuid[0:7] = 0x00
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

6.2.2 Special purpose register (SPR) summary

PowerISA 2.06 and implementation-specific SPRs for the e200z215An3 core are listed in *Table 13*. All registers are 32 bits in size. Register bits are numbered from bit 0 to bit 31 (most-significant to least-significant). 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 13*.

Table 13. Special purpose registers

Mnemonic	Name	SPR number	Access	Privileged	e200z specific
XER	Integer Exception Register	1	R/W	No	No
LR	Link Register	8	R/W	No	No
CTR	Count Register	9	R/W	No	No
SRR0	Save/Restore Register 0	26	R/W	Yes	No
SRR1	Save/Restore Register 1	27	R/W	Yes	No
PID0	Process ID Register	48	R/W	Yes	No
CSRR0	Critical Save/Restore Register 0	58	R/W	Yes	No
CSRR1	Critical Save/Restore Register 1	59	R/W	Yes	No
DEAR	Data Exception Address Register	61	R/W	Yes	No

Table 13. Special purpose registers(Continued)

Mnemonic	Name	SPR number	Access	Privileged	e200z specific
ESR	Exception Syndrome Register	62	R/W	Yes	No
IVPR	Interrupt Vector Prefix Register	63	R/W	Yes	No
USPRG0 (VRSAVE)	User SPR General 0 (renamed to VRSAVE in *PowerISA 2.06*)	256	R/W	No	No
SPRG0	SPR General 0	272	R/W	Yes	No
SPRG1	SPR General 1	273	R/W	Yes	No
SPRG2	SPR General 2	274	R/W	Yes	No
SPRG3	SPR General 3	275	R/W	Yes	No
PIR	Processor ID Register	286	R/W	Yes	No
PVR	Processor Version Register	287	Read-only	Yes	No
DBSR	Debug Status Register	304	Read/Clear (1)	Yes	No
DBCR0	Debug Control Register 0	308	R/W	Yes	No
DBCR1	Debug Control Register 1	309	R/W	Yes	No
DBCR2	Debug Control Register 2	310	R/W	Yes	No
IAC1	Instruction Address Compare 1	312	R/W	Yes	No
IAC2	Instruction Address Compare 2	313	R/W	Yes	No
IAC3	Instruction Address Compare 3	314	R/W	Yes	No
IAC4	Instruction Address Compare 4	315	R/W	Yes	No
DAC1	Data Address Compare 1	316	R/W	Yes	No
DAC2	Data Address Compare 2	317	R/W	Yes	No
DVC1	Data Value Compare 1(2)	318	R/W	Yes	No
DVC2	Data Value Compare 2(2)	319	R/W	Yes	No
SPEFSCR	EFP APU status and control register	512	R/W	No	No
L1CFG0	L1 cache configuration register 0	515	Read-only	No	Yes
L1CFG1	L1 cache configuration register 1	516	Read-only	No	Yes
NPIDR	Nexus 3 Process ID register	517	R/W	No	Yes
DBCR3	Debug control register 3	561	R/W	Yes	Yes
DBCR4	Debug control register 4	563	R/W	Yes	Yes
DBCR5	Debug control register 5	564	R/W	Yes	Yes
IAC5	Instruction Address Compare 5	565	R/W	Yes	Yes
IAC6	Instruction Address Compare 6	566	R/W	Yes	Yes
IAC7	Instruction Address Compare 7	567	R/W	Yes	Yes
IAC8	Instruction Address Compare 8	568	R/W	Yes	Yes
MCSRR0	Machine Check Save/Restore Register 0	570	R/W	Yes	Yes

Table 13. Special purpose registers(Continued)

Mnemonic	Name	SPR number	Access	Privileged	e200z specific
MCSRR1	Machine Check Save/Restore Register 1	571	R/W	Yes	Yes
MCSR	Machine Check Syndrome Register	572	Read/Clear (3)	Yes	Yes
MCAR	Machine Check Address Register	573	R/W	Yes	Yes
DSRR0	Debug save/restore register 0	574	R/W	Yes	Yes
DSRR1	Debug save/restore register 1	575	R/W	Yes	Yes
DDAM	Debug Data Acquisition Messaging register	576	R/W	No	Yes
DAC3	Data Address Compare 3	592	R/W	Yes	Yes
DAC4	Data Address Compare 4	593	R/W	Yes	Yes
DBCR7	Debug control register 7	596	R/W	Yes	Yes
DBCR8	Debug control register 8	597	R/W	Yes	Yes
DDEAR	Debug Data Effective Address register	600	R/W	Yes	Yes
DVC1U(4)	Data Value Compare 1 Upper	601	R/W	Yes	No
DVC2U(4)	Data Value Compare 2 Upper	602	R/W	Yes	No
DBCR6	Debug control register 6	603	R/W	Yes	Yes
EDBRAC0	External debug resource allocation control register 0	638	Read-only	Yes	Yes
DMEMCFG0	Local Data Memory config register 0	694	Read-only	Yes	Yes
IMEMCFG0	Local Instruction Memory config register 0	695	Read-only	Yes	Yes
DEVENT	Debug Event register	975	R/W	No	Yes
HID0	Hardware implementation dependent reg 0	1008	R/W	Yes	Yes
HID1	Hardware implementation dependent reg 1	1009	R/W	Yes	Yes
BUCSR	Branch Unit Control and Status Register	1013	R/W	Yes	Yes
MMUCFG	MMU/MPU configuration register	1015	Read-only	Yes	Yes
SVR	System Version Register	1023	Read-only	Yes	Yes

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

6.3 System modules

6.3.1 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, refer *Chapter 13: System Integration Unit Lite2 (SIUL2)*.

6.3.2 Crossbar switch configuration

This device contains one crossbar switch module. The following sections explain how the crossbar switch is configured on this device.

6.3.2.1 Crossbar switch (XBAR_0) 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 6. Crossbar switch integration

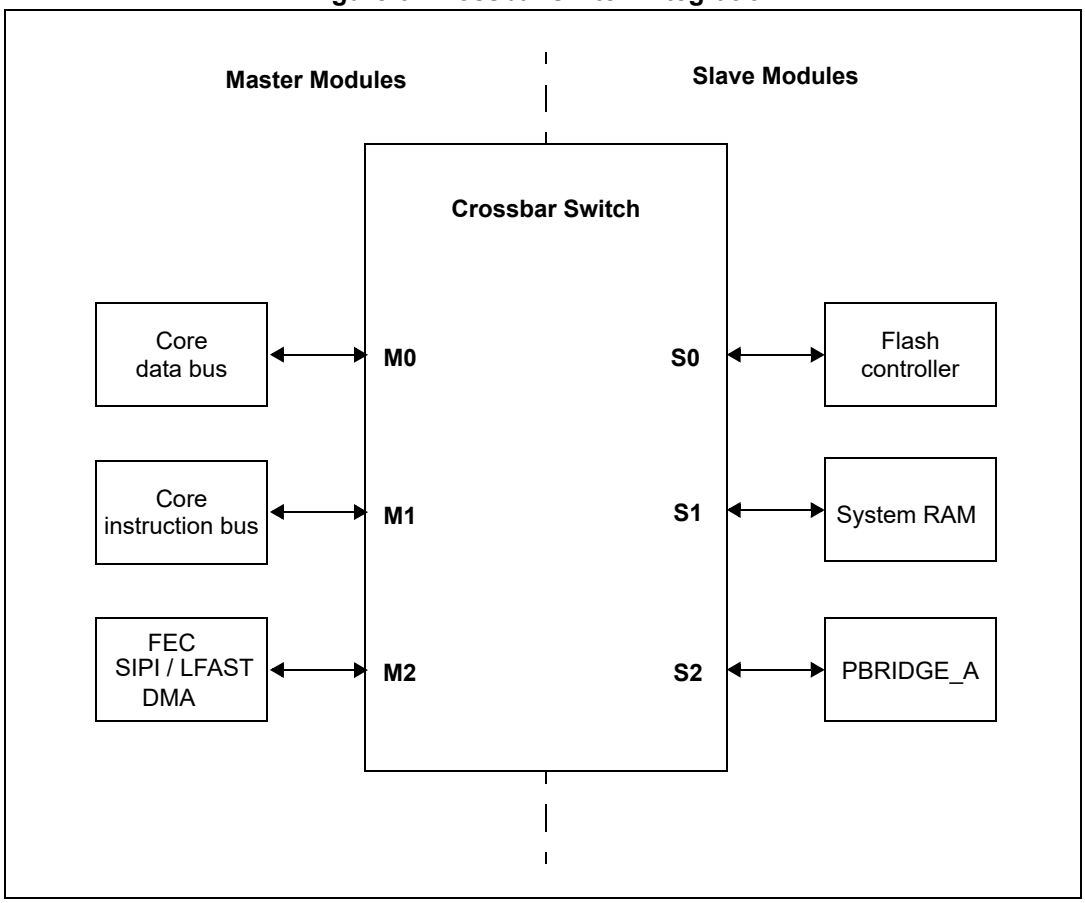


Table 14. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	*Chapter 14: Crossbar Switch (XBAR)*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 21: Clocking*
Crossbar switch master	Core	*Section 6.2: Core module*
Crossbar switch slave	Flash memory controller	*Section 3.3.1: Flash memory controller*
Crossbar switch slave	System RAM	*Section 3.2.1: System SRAM*

6.3.2.1.1 Crossbar switch master assignments

This device contains three masters connected to the crossbar switch.

Table 15 contains the master assignments.

Table 15. Crossbar switch master assignments

Master module	Physical master port number
Core data bus	0
Core instruction bus	1
FEC, SIPI / LFAST, and DMA	2

6.3.2.1.2 Crossbar switch slave assignments

This device contains three slaves connected to the crossbar switch.

Table 16 contains the slave assignments.

Table 16. Crossbar switch slave assignments

Slave module	Physical slave port number
Flash memory controller port 0	0
System RAM	1
PBRIDGE_A	2

6.3.2.1.3 PRS register reset values

The XBAR_0 PRS*n* registers reset to 0000_0021h.

6.3.3 System Memory Protection Unit (SMPU) configuration

This chip contains one System Memory Protection Unit (SMPU): SMPU_0.

The following sections explain how SMPU_0 is configured on this chip.

6.3.3.1 SMPU_0 configuration

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

Figure 7. SMPU_0 configuration

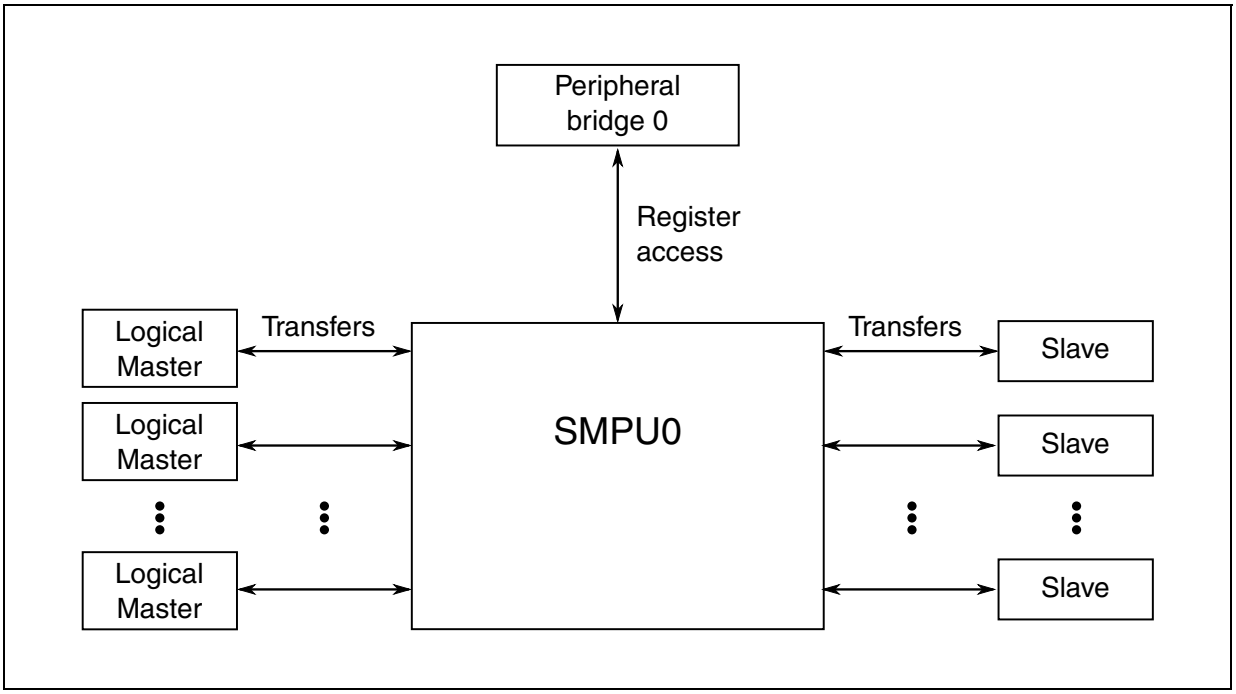


Table 17. Reference links to related information

Topic	Related module	Reference
Full description	System Memory Protection Unit (SMPU)	*Chapter 16: System Memory Protection Unit (SMPU)*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 21: Clocking*
Power management	—	*Chapter 9: Power management*
Logical masters	—	*Section 6.3.3.1.1: SMPU_0 logical bus master assignments*
Region descriptors	—	*Section 6.3.3.1.2: Number of region descriptors supported by SMPU_0*
RGD_WORD2 formats	—	*Section 6.3.3.1.3: RGD_WORD2 formats supported by SMPU_0*

6.3.3.1.1 SMPU_0 logical bus master assignments

Table 18 contains the logical bus master assignments for the SMPU_0.

Table 18. SMPU_0 logical bus master assignments

SMPU logical bus master number	Bus master
0	Core
1	Reserved
2	Reserved
3	DMA
4	Ethernet
5	Reserved
6	Reserved
7	SIPI/LFAST
8	Core Debug
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved

6.3.3.1.2 Number of region descriptors supported by SMPU_0

This instance of the SMPU module supports up to 12 region descriptors.

6.3.3.1.3 RGD_WORD2 formats supported by SMPU_0

For RGD_WORD2, this instance of the SMPU module supports the FMT0 and FMT1 formats. Refer to *Section 16.4.2.7: Region Descriptor n, Word 2 Format 0 (SMPU0_RGD_WORD2_FMT0)* and *Section 16.4.2.9: Region Descriptor n, Word 3 (SMPU0_RGDn_WORD3)* for further details.

6.3.4 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.

This device contains one peripheral bridge.

Figure 8. Peripheral bridge configuration

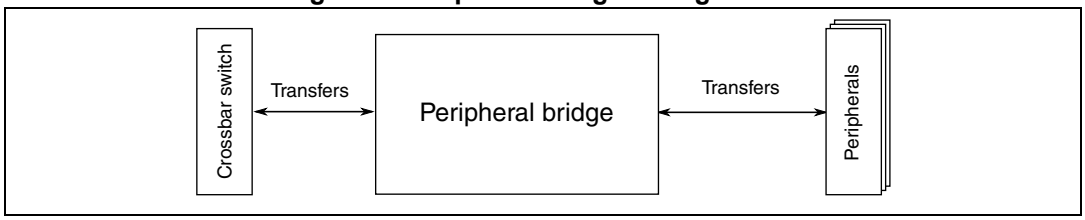


Table 19. Reference links to related information

Topic	Related module	Reference
Full description	Peripheral bridge	*Chapter 15: Peripheral Bridge*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 21: Clocking*
Crossbar switch	Crossbar switch	*Section 6.3.2: Crossbar switch configuration*

6.3.4.1 Memory maps

The peripheral bridge is 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.

6.3.4.2 MPR registers

The peripheral bridge supports up to 16 crossbar switch masters, each assigned to a MPROT*n* field in the MPRA-B registers. However, fewer are supported on this device. See *Section 6.3.2: Crossbar switch configuration* for details of the master port assignments for this device.

6.3.4.3 PACR/OPACR registers

The peripheral bridge supports 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.

6.3.5 Platform Configuration Module (PCM)

The Platform Configuration Module contains the miscellaneous configuration register: IAHB Burst Enable 2 (IAHB_BE2) register

Currently, the configuration registers are related to the operation of the intelligent bus bridging gasket. The module is mapped to PBRIDGE on-platform slot 10 with a base address of 0xFC02_8000.

6.3.5.1 IAHB Burst Enable 2 (IAHB_BE2) register

Offset: 08h 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	RSV	RSV	RSV	0	0	0	0	0	RSV	RSV	RSV
W																
Reset	0	0	0	0	0	—	—	—	0	0	0	0	0	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	RSV	RSV	RSV	0	0	0	0	0	PRE_M2	BRE_M2	BWE_M2
W																
Reset	0	0	0	0	0	—	—	—	0	0	0	0	0	1	1	1

Figure 9. IAHB Burst Enable 2 (IAHB_BE2) register

Table 20. IAHB_BE2 field descriptions

Field	Description
0–4	Reserved
5–7 RSV	Reserved, undefined
8–12	Reserved
13–15 RSV	Reserved, undefined
16–20	Reserved
21–23 RSV	Reserved, undefined
24–28	Reserved
29 PRE_M2	Pending read enable (PRE) M2 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_M2	Burst read enable (BRE) M2 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_M2	Burst write enable (BWE) M2 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. **Note:** This setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

6.3.6 Interrupt Controller (INTC) configuration

6.3.6.1 INTC implemented registers

Table 21 shows the registers implemented on the INTC on this chip.

Table 21. INTC implemented registers

Address offset	Register
000h	INTC Block Configuration Register (INTC_BCR)
010h	INTC Current Priority Register for Processor (INTC_CPR0)
014h	Reserved
018h	Reserved
020h	INTC Interrupt Acknowledge Register for Processor (INTC_IACKR0)
024h	Reserved
028h	Reserved

Table 21. INTC implemented registers(Continued)

Address offset	Register
030h	INTC End Of Interrupt Register for Processor (INTC_EOIR0)
034h	Reserved
038h	Reserved
040h–05Ch	INTC Software Set/Clear Interrupt Register 0 (INTC_SSCIR0) – INTC Software Set/Clear Interrupt Register 31 (INTC_SSCIR31)
05Ch–85Ch	INTC Priority Select Register 0 (INTC_PSR0) – INTC Priority Select Register 1023 (INTC_PSR1023)

6.3.6.2 Interrupt sources

Table 22 defines the interrupt sources for the INTC on this chip. All IRQ numbers not specifically listed are not used.

Table 22. Interrupt sources

IRQ #	Offset	Source description	Source name
0	1000h	Software settable flag 0	SSCIR0_3[CLR0]
1	1004h	Software settable flag 1	SSCIR0_3[CLR1]
2	1008h	Software settable flag 2	SSCIR0_3[CLR2]
3	100Ch	Software settable flag 3	SSCIR0_3[CLR3]
4	1010h	Software settable flag 4	SSCIR4_7[CLR4]
5	1014h	Software settable flag 5	SSCIR4_7[CLR5]
6	1018h	Software settable flag 6	SSCIR4_7[CLR6]
7	101Ch	Software settable flag 7	SSCIR4_7[CLR7]
8	1020h	Software settable flag 8	SSCIR8_11[CLR8]
9	1024h	Software settable flag 9	SSCIR8_11[CLR9]
10	1028h	Software settable flag 10	SSCIR8_11[CLR10]
11	102Ch	Software settable flag 11	SSCIR8_11[CLR11]
12	1030h	Software settable flag 12	SSCIR12_15[CLR12]
13	1034h	Software settable flag 13	SSCIR12_15[CLR13]
14	1038h	Software settable flag 14	SSCIR12_15[CLR14]
15	103Ch	Software settable flag 15	SSCIR12_15[CLR15]
16	1040h	Software settable flag 16	SSCIR16_19[CLR16]
17	1044h	Software settable flag 17	SSCIR16_19[CLR17]
18	1048h	Software settable flag 18	SSCIR16_19[CLR18]
19	104Ch	Software settable flag 19	SSCIR16_19[CLR19]
20	1050h	Software settable flag 20	SSCIR20_23[CLR20]
21	1054h	Software settable flag 21	SSCIR20_23[CLR21]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
22	1058h	Software settable flag 22	SSCIR20_23[CLR22]
23	105Ch	Software settable flag 23	SSCIR20_23[CLR23]
24	1060h	Software settable flag 24	SSCIR24_27[CLR24]
25	1064h	Software settable flag 25	SSCIR24_27[CLR25]
26	1068h	Software settable flag 26	SSCIR24_27[CLR26]
27	106Ch	Software settable flag 27	SSCIR24_27[CLR27]
28	1070h	Software settable flag 28	SSCIR28_31[CLR28]
29	1074h	Software settable flag 29	SSCIR28_31[CLR29]
30	1078h	Software settable flag 30	SSCIR28_31[CLR30]
31	107Ch	Software settable flag 31	SSCIR28_31[CLR31]
34	1088h	Software watchdog timer 2 (SWT_2) Timeout	SWT_2_IR[TIF]
35	108Ch	Software watchdog timer 3 (SWT_3) Timeout	SWT_3_IR[TIF]
44	10B0h	System Timer 2 Channel 0 match	STM_2_CIR0[CIF]
45	10B4h	System Timer 2 Channel 1 match	STM_2_CIR1[CIF]
46	10B8h	System Timer 2 Channel 2 match	STM_2_CIR2[CIF]
47	10BCh	System Timer 2 Channel 3 match	STM_2_CIR3[CIF]
52	10D0h	eDMA Combined Channel Error Flags 31–0	eDMA Channel Error Flags 31–0
53	10D4h	eDMA Channel interrupt 0	DMA_INTL[INT0]
54	10D8h	eDMA Channel interrupt 1	DMA_INTL[INT1]
55	10DCh	eDMA Channel interrupt 2	DMA_INTL[INT2]
56	10E0h	eDMA Channel interrupt 3	DMA_INTL[INT3]
57	10E4h	eDMA Channel interrupt 4	DMA_INTL[INT4]
58	10E8h	eDMA Channel interrupt 5	DMA_INTL[INT5]
59	10ECh	eDMA Channel interrupt 6	DMA_INTL[INT6]
60	10F0h	eDMA Channel interrupt 7	DMA_INTL[INT7]
61	10F4h	eDMA Channel interrupt 8	DMA_INTL[INT8]
62	10F8h	eDMA Channel interrupt 9	DMA_INTL[INT9]
63	10FCh	eDMA Channel interrupt 10	DMA_INTL[INT10]
64	1100h	eDMA Channel interrupt 11	DMA_INTL[INT11]
65	1104h	eDMA Channel interrupt 12	DMA_INTL[INT12]
66	1108h	eDMA Channel interrupt 13	DMA_INTL[INT13]
67	110Ch	eDMA Channel interrupt 14	DMA_INTL[INT14]
68	1110h	eDMA Channel interrupt 15	DMA_INTL[INT15]
181	12D4	PlatformFlash ECC error SBC	ACP_ERM_ECC_single_bit_correct
182	12D8	PlatformRam ECC error SBC	ACP_ERM_ECC_multi_bit_detect

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
185	12E4h	Flash controller Prog/Erase/Suspend IRQ_0	MCR[DONE]
218	1368h	Ethernet Transmit frame interrupt	EIR[TXF]
219	136Ch	Ethernet Receive frame interrupt	EIR[RXF]
220	1370h	Ethernet combined interrupt	EIR[HBERR] EIR[BABR] EIR[BABT] EIR[GRA] EIR[TXB] EIR[RXB] EIR[MII] EIR[EBERR] EIR[LC] EIR[RL] EIR[UN]
226	1388h	PIT0 Channel 0	PIT_0_TFLG0[TIF]
227	138Ch	PIT0 Channel 1	PIT_0_TFLG1[TIF]
228	1390h	PIT0 Channel 2	PIT_0_TFLG2[TIF]
229	1394h	PIT0 Channel 3	PIT_0_TFLG3[TIF]
240	13C0h	PIT1 Channel 0	PIT_1_TFLG0[TIF]
241	13C4h	PIT1 Channel 1	PIT_1_TFLG1[TIF]
242	13C8h	Crystal oscillator (XOSC) clock interrupt	XOSC_CTL[I_OSC]
243	13CCh	SIUL2 Combined External Interrupt 0 (= OR of REQ5, REQ0)	SIUL2_DISR0[EIF5] SIUL2_DISR0[EIF0]
244	13D0h	SIUL2 Combined External Interrupt 1 (REQ10)	SIUL2_DISR0[EIF10]
251	13ECh	MC_ME Safe Mode interrupt	ME_IS[I_SAFE]
252	13F0h	MC_ME Mode Transition interrupt	ME_IS[I_MTC]
253	13F4h	MC_ME Invalid Mode interrupt	ME_IS[I_IMODE]
254	13F8h	MC_ME Invalid Mode Configuration	ME_IS[I_ICONF]
255	13FCh	MC_RGM 0	MC_RGM Functional and destructive reset alternate event interrupt
259	140Ch	DSPI_0 combined overrun interrupt requests: – Transmit FIFO underflow – Receive FIFO overflow	DSPI_0_SR[TFUF] | DSPI_0_SR[RFOF]
260	1410h	DSPI_0 transmit FIFO end-of-queue flag	DSPI_0_SR[EOQF]
261	1414h	DSPI_0 Tx FIFO fill request flag	DSPI_0_SR[TFFF]
262	1418h	DSPI_0 transfer complete flag	DSPI_0_SR[TCF]
263	141Ch	DSPI_0 Rx FIFO drain request flag	DSPI_0_SR[RFDF]
264	1420	DSPI_0 command transfer complete flag	DSPI_0_SR[CMD_TCF]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
265	1424	DSPI_0 transmit FIFO fill flag	DSPI_0_SR[CMDFFF]
266	1428h	DSPI_0 combined DSI and SPI Parity Error Flags	DSPI_0_SR[SPEF]
295	149Ch	DSPI_4 combined overrun interrupt requests: – FIFO underflow – Receive FIFO overflow	DSPI_4_SR[TFUF] | DSPI_4_SR[RFOF]
296	14A0h	DSPI_4 transmit FIFO end-of-queue flag	DSPI_4_SR[EOQF]
297	14A4h	DSPI_4 Tx FIFO fill request flag	DSPI_4_SR[TFFF]
298	14A8h	DSPI_4 transfer complete flag	DSPI_4_SR[TCF]
299	14ACh	DSPI_4 Rx FIFO drain request flag	DSPI_4_SR[RFDF]
300	14B0h	DSPI_4 SPI Frame Transfer Complete Flag	DSPI_4_SR[SPITCF]
301	14B4h	DSPI_4 DSI Frame Transfer Complete Flag	DSPI_4_SR[DSITCF]
302	14B8h	DSPI_4 combined DSI and SPI Parity Error Flags	DSPI_4_SR[SPEF] | DSPI_4_SR[DPEF]
303	14BCh	DSPI_4 DSI Data Received with Active Bits	DSPI_4_SR[DDIF]
376	15E0h	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	15E4h	LINFlexD_0 Transmission combined interrupt: – Header Received – Data Transmitted – Data Buffer Empty	LINFlexD_0_LINIER[HRIE] | LINFlexD_0_LINIER[DTIE] | LINFlexD_0_LINIER[DBEIE]
378	15E8h	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]
380	15F0h	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]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
381	15F4h	LINFlexD_1 Transmission combined interrupt: – Header Received – Data Transmitted – Data Buffer Empty	LINFlexD_1_LINIER[HRIE] | LINFlexD_1_LINIER[DTIE] | LINFlexD_1_LINIER[DBEIE]
382	15F8h	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]
432	16C0h	LINFlexD_14 Reception combined interrupt – Header Received – Data Reception complete – Data Buffer Full – Wakeup – LIN State (Sync Del, Sync Field, Identifier Field or Checksum Field)	LINFlexD_14_LINIER[HRIE] | LINFlexD_14_LINIER[DRIE] | LINFlexD_14_LINIER[DBFIE] | LINFlexD_14_LINIER[WUIE] | LINFlexD_14_LINIER[LSIE]
433	16C4h	LINFlexD_14 Transmission combined interrupt: – Header Received – Data Transmitted – Data Buffer Empty	LINFlexD_14_LINIER[HRIE] | LINFlexD_14_LINIER[DTIE] | LINFlexD_14_LINIER[DBEIE]
434	16C8h	LINFlexD_14 Error combined interrupt: – Stuck at zero – Output Compare – Bit Error – Checksum Error – Header Error – Frame Error – Buffer Overrun – UART Timeout Error	LINFlexD_14_LINIER[SZIE] | LINFlexD_14_LINIER[OCIE] | LINFlexD_14_LINIER[BEIE] | LINFlexD_14_LINIER[CEIE] | LINFlexD_14_LINIER[HEIE] | LINFlexD_14_LINIER[FEIE] | LINFlexD_14_LINIER[BOIE] | LINFlexD_14_LINIER[TOIE]
477	1774h	Power Management Controller Supply Gauge Low Voltage Detect and High Voltage Detect Interrupts	GR_S[VD15] | GR_S[VD14] | GR_S[VD13] | GR_S[VD12] | GR_S[VD10] | GR_S[VD9] | GR_S[VD7] | GR_S[VD4] | GR_S[VD3]
478	1778h	Power Management Controller Temperature Sensor interrupts	EPR_TD[TEMP_3] | EPR_TD[TEMP_2] | EPR_TD[TEMP_0]
480	1780h	Loss-of-lock interrupt request for PLL0	PLL0SR[LOLF]
484	1790h	PLL_0_1	PLL0SR[EXTPDF]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
528	1840h	SAR ADC0 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) – Add WDGxL (watchDog Low int) – WDGxH (watchDog High int)	SARADC_0_ICIPR0[EOC_CH[7]] | SARADC_0_ICIPR0[EOC_CH[6]] | SARADC_0_ICIPR0[EOC_CH[5]] | SARADC_0_ICIPR0[EOC_CH[4]] I SARADC_0_ICIPR0[EOC_CH[8]] I SARADC_0_ICIPR0[EOC_CH[0]] I SARADC_0_ICIPR0[EOC_CH[1]] I SARADC_0_ICIPR0[EOC_CH[2]] I SARADC_0_ICIPR0[EOC_CH[3]]
532	1850h	SAR ADC4 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) – Add WDGxL (watchDog Low int) – WDGxH (watchDog High int)	SARADC_4_ICIPR0[EOC_CH[61]]| ..... SARADC_4_ICIPR0[EOC_CH[55]]| SARADC_4_ICIPR0[EOC_CH[54]]| SARADC_4_ICIPR0[EOC_CH[53]]| SARADC_4_ICIPR0[EOC_CH[52]]
543	187Ch	SAR_ADCB 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) – Add WDGxL (watchDog Low int) – WDGxH (watchDog High int)	SARADC_B_ICIPR0[EOC_CH[31] | SARADC_B_ICIPR0[EOC_CH[30] | ... | SARADC_B_ICIPR0[EOC_CH[1] | SARADC_B_ICIPR0[EOC_CH[0]] | SARADC_B_ICIPR1[EOC_CH[61]] | ... | SARADC_B_ICIPR1[EOC_CH[56]] | SARADC_B_ICIPR1[EOC_CH[53]] | ... | SARADC_B_ICIPR1[EOC_CH[0]]
547	188Ch	SD ADC3 combined interrupt: – Data FIFO empty – Data FIFO full – Data FIFO overrun	SDADC_3_SFR[DFEF] | SDADC_3_SFR[DFORF] | SDADC_3_SFR[DFFF]
558	18B8h	SENT_0 Fast messages combined interrupt: OR of all enabled Fast message Ready flags	SENT_0_FMSG_RDY[0] | SENT_0_FMSG_RDY[1] | SENT_0_FMSG_RDY[2] | SENT_0_FMSG_RDY[3]
559	18BCh	SENT_0 Slow messages combined interrupt: OR of all enabled Slow message Ready flags	SENT_0_SMSG_RDY[0] | SENT_0_SMSG_RDY[1] | SENT_0_SMSG_RDY[2] | SENT_0_SMSG_RDY[3]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
560	18C0h	SENT_0 Error combined interrupt: OR of following events: – Fast message DMA underflow – Slow message DMA underflow – Channels 0 to 3: - Successive calibration check resynchronized error - Calibration variation 20 – 25% - Slow message overflow - Fast message overflow - Number of negative edges error - Checksum error in fast message - Checksum error in slow message - Nibble data value error - Calibration pulse length error - Calibration pulse length ratio to message length error	SENT_0_GBL_STATUS[FMDU] | SENT_0_GBL_STATUS[SMDU] | SENT_0_CHn_STATUS[4:7] | SENT_0_CHn_STATUS[9:15]
564	18D0h	SENT_0 Channel 0 Fast message interrupt	SENT_0_FMSG_RDY[0]
565	18D4h	SENT_0 Channel 0 Slow message interrupt	SENT_0_SMSG_RDY[0]
566	18D8h	SENT_0 Channel 0 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_0_CH0_STATUS[4:7] | SENT_0_CH0_STATUS[9:15]
567	18DCh	SENT_0 Channel 1 Fast message interrupt	SENT_0_FMSG_RDY[1]
568	18E0h	SENT_0 Channel 1 Slow message interrupt	SENT_0_SMSG_RDY[1]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
569	18E4h	SENT_0 Channel 1 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_0_CH1_STATUS[4:7] | SENT_0_CH1_STATUS[9:15]
570	18E8h	SENT_0 Channel 2 Fast message interrupt	SENT_0_FMSG_RDY[2]
571	18ECh	SENT_0 Channel 2 Slow message interrupt	SENT_0_SMSG_RDY[2]
572	18F0h	SENT_0 Channel 2 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_2_CH2_STATUS[4:7] | SENT_2_CH2_STATUS[9:15]
573	18F4h	SENT_0 Channel 3 Fast message interrupt	SENT_0_FMSG_RDY[3]
574	18F8h	SENT_0 Channel 3 Slow message interrupt	SENT_0_SMSG_RDY[3]
575	18FCh	SENT_0 Channel 3 combined Error interrupt: – Successive calibration check resynchronized error – Calibration variation 20 – 25% – Slow message overflow – Fast message overflow – Number of negative edges error – Checksum error in fast message – Checksum error in slow message – Nibble data value error – Calibration pulse length error – Calibration pulse length ratio to message length error	SENT_0_CH3_STATUS[4:7] | SENT_0_CH3_STATUS[9:15]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
654	1A38h	SIPI combined Error Interrupt: – Channels 0 to 3 Timeout error – Channels 0 to 3 transaction ID error – Channels 0 to 3 acknowledge error	SIPI_ERR[TOEn] | SIPI_ERR[TIDEn] | SIPI_ERR[ACKEn] n = [0..3]
655	1A3Ch	SIPI CRC Error interrupt	SIPI_SR[GCRCE)
656	1A40h	SIPI Channel 0 Reception combined interrupt: – Read answer received – Acknowledge received	SIPI_CSR0[RAR] | SIPI_CSR0[ACKR]
657	1A44h	SIPI Channel 1 Reception combined interrupt: – Read answer received – Acknowledge received	SIPI_CSR1[RAR] | SIPI_CSR1[ACKR]
658	1A48h	SIPI Channel 2 Reception combined interrupt: – Read answer received – Acknowledge received	SIPI_CSR2[RAR] | SIPI_CSR2[ACKR]
659	1A4Ch	SIPI Channel 3 Reception combined interrupt: – Read answer received – Acknowledge received	SIPI_CSR3[RAR] | SIPI_CSR3[ACKR]
660	1A50h	SIPI combined Event Interrupt: – Channel 0 to 3 trigger events – Maximum count reached (streaming mode) event	SIPI_SR[TE0] | SIPI_SR[TE1] |SIPI_SR[TE2] |SIPI_SR[TE3] |SIPI_SR[MCR]
661	1A54h	LFAST_0 Transmit Complete combined interrupt: – Ping response frame transmitted – Unsolicited Frame transmitted – ICLC Frame transmitted – Data frame transmitted	LFAST_0_TISR[TXPNGF] | LFAST_0_TISR[TXUNSF] | LFAST_0_TISR[TXICLCF] | LFAST_0_TISR[TXDTF]
662	1A58h	LFAST_0 Transmit Error combined interrupt: – Transmit interface not enabled error – Transmit data FIFO overflow error	LFAST_0_TISR[TXIEF] | LFAST_0_TISR[TXOVF]
663	1A5Ch	LFAST_0 Receive combined interrupt: – Frame with CTS bit low received – Data frame received – Unsolicited Frame received	LFAST_0_RISR[RXCTSF] | LFAST_0_RISR[RXDF] | LFAST_0_RISR[RXUNSF]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
664	1A60h	LFAST_0 Receive Error combined interrupt: – Unsolicited frame register overflow – Rx Data FIFO Minimum Threshold reached – Rx Data FIFO Max Threshold reached – Rx Data FIFO Underflow – Rx Data FIFO Overflow – Frame with unsupported frame size received – Invalid ICLC code Received – Invalid Logical Channel Type	LFAST_0_RISR[RXUOF] | LFAST_0_RISR[RXMNF] | LFAST_0_RISR[RXMXF] | LFAST_0_RISR[RXUFF] | LFAST_0_RISR[RXOFF] | LFAST_0_RISR[RXSZF] | LFAST_0_RISR[RXICF] | LFAST_0_RISR[RXLCEF]
665	1A64h	LFAST_0 ICLC receive combined interrupt: – Ping Frame Response failed – Ping Frame Response successful – ICLC Ping Frame Request received – ICLC frame for Test mode off received – ICLC frame for Loopback On received – ICLC frame for Clk Test mode received – ICLC frame for LFAST Slaves Tx Interface – Disable received – ICLC frame for LFAST Slaves Tx Interface Enable received – ICLC frame for LFAST Slaves Rx Interface fast mode switch received – ICLC frame for LFAST Slaves Rx Interface slow mode switch received – ICLC frame for LFAST Slaves Tx Interface fast mode switch received – ICLC frame for LFAST Slaves Tx Interface slow mode switch received – ICLC frame for PLL OFF received – ICLC frame for PLL ON received	LFAST_0_RIISR[ICPFF] | LFAST_0_RIISR[ICPSF] | LFAST_0_RIISR[ICPRF] | LFAST_0_RIISR[ICTOF] | LFAST_0_RIISR[ICLPF] | LFAST_0_RIISR[ICCTF] | LFAST_0_RIISR[ICTDF] | LFAST_0_RIISR[ICTEF] | LFAST_0_RIISR[ICRFF] | LFAST_0_RIISR[ICRSF] | LFAST_0_RIISR[ICTFF] | LFAST_0_RIISR[ICTSF] | LFAST_0_RIISR[ICPOFF] | LFAST_0_RIISR[ICPONF]
674	1A88h	JTAG Master combined interrupt: – Idle interrupt – Sequence Processing Unit interrupt	JTAGM_SR[SPU_INT] | JTAGM_SR[Idle]
675	1A8Ch	JTAG Data Communication combined interrupt: – JDC Input data received – JDC output data sent	JDC_MSR[JIN_INT] | JDC_MSR[JOUT_INT]
688	1AC0h	M_CAN_1 Line0 combined interrupt	(M_CAN_1_IR[0] & ~M_CAN_1_ILS[0])| ... | (M_CAN_1_IR[11] & ~M_CAN_1_ILS[11]) | (M_CAN_1_IR[13] & ~M_CAN_1_ILS[13]) | … | (M_CAN_1_IR[31] & ~M_CAN_1_ILS[31])
689	1AC4h	M_CAN_1 Line1 combined interrupt	(M_CAN_1_IR[0] & M_CAN_1_ILS[0])| ... | (M_CAN_1_IR[11] & M_CAN_1_ILS[11]) | (M_CAN_1_IR[13] & M_CAN_1_ILS[13]) | … | (M_CAN_1_IR[31] & M_CAN_1_ILS[31])

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
690	1AC8h	M_CAN_2 Line0 combined interrupt	(M_CAN_2_IR[0] & ~M_CAN_2_ILS[0])| ... | (M_CAN_2_IR[11] & ~M_CAN_2_ILS[11]) | (M_CAN_2_IR[13] & ~M_CAN_2_ILS[13]) | … | (M_CAN_2_IR[31] & ~M_CAN_2_ILS[31])
691	1ACCh	M_CAN_2 Line1 combined interrupt	(M_CAN_2_IR[0] & M_CAN_2_ILS[0])| ... | (M_CAN_2_IR[11] & M_CAN_2_ILS[11]) | (M_CAN_2_IR[13] & M_CAN_2_ILS[13]) | … | (M_CAN2__IR[31] & M_CAN_2_ILS[31])
706	1B08h	GTM AEI Shared interrupt	GTM_AEI_IRQ AEI
707	1B0Ch	GTM ARU New Data0 transferred interrupt	GTM_ARU_IRQ[0]
708	1B10h	GTM ARU New Data1 transferred interrupt	GTM_ARU_IRQ[1]
709	1B14h	GTM ARU access acknowledge interrupt	GTM_ARU_IRQ[2]
711	1B1Ch	GTM compare shared module interrupt	GTM_CMP_IRQ
712	1B20h	GTM Sensor pattern evaluation 0 interrupt	GTM_SPE0_IRQ
713	1B24h	GTM Sensor pattern evaluation 1 interrupt	GTM_SPE1_IRQ
714	1B28h	GTM Parameter Storage Module 0 Channel 0 interrupt	GTM_PSM0_IRQ[0]
715	1B2Ch	GTM Parameter Storage Module 0 Channel 1 interrupt	GTM_PSM0_IRQ[1]
716	1B30h	GTM Parameter Storage Module 0 Channel 2 interrupt	GTM_PSM0_IRQ[2]
717	1B34h	GTM Parameter Storage Module 0 Channel 3 interrupt	GTM_PSM0_IRQ[3]
718	1B38h	GTM Parameter Storage Module 0 Channel 4 interrupt	GTM_PSM0_IRQ[4]
719	1B3Ch	GTM Parameter Storage Module 0 Channel 5 interrupt	GTM_PSM0_IRQ[5]
720	1B40h	GTM Parameter Storage Module 0 Channel 6 interrupt	GTM_PSM0_IRQ[6]
721	1B44h	GTM Parameter Storage Module 0 Channel 7 interrupt	GTM_PSM0_IRQ[7]
722	1B48h	DPLL_DCGI: DPLL direction change interrupt	GTM_DPLL_IRQ[0]
723	1B4Ch	DPLL_EDI: DPLL enable or disable interrupt	GTM_DPLL_IRQ[1]
724	1B50h	DPLL_TINI: DPLL TRIG. min. hold time (THMI) viol. detected	GTM_DPLL_IRQ[2]
725	1B54h	DPLL_TAXI: DPLL TRIG. max. hold time (THMA) viol. detected	GTM_DPLL_IRQ[3]
726	1B58h	DPLL_SISI: DPLL STATE inactive slope detected	GTM_DPLL_IRQ[4]
727	1B5Ch	DPLL_TISI: DPLL TRIGGER inactive slope detected	GTM_DPLL_IRQ[5]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
728	1B60h	DPLL_MSI: DPLL Missing STATE interrupt	GTM_DPLL_IRQ[6]
729	1B64h	DPLL_MTI: DPLL Missing TRIGGER interrupt	GTM_DPLL_IRQ[7]
730	1B68h	DPLL_SASI: DPLL STATE active slope detected	GTM_DPLL_IRQ[8]
731	1B6Ch	DPLL_TASI: DPLL TRIG. active slope det. while NTI_CNT is 0	GTM_DPLL_IRQ[9]
732	1B70h	DPLL_PWI: DPLL Plausibility window (PVT) viol. int. of TRIG.	GTM_DPLL_IRQ[10]
733	1B74h	DPLL_W2I: DPLL Write access to RAM region 2 interrupt	GTM_DPLL_IRQ[11]
734	1B78h	DPLL_W1I: DPLL Write access to RAM region 1b or 1c int.	GTM_DPLL_IRQ[12]
735	1B7Ch	DPLL_GL1I: DPLL Get of lock interrupt for SUB_INC1	GTM_DPLL_IRQ[13]
736	1B80h	DPLL_LL1I: DPLL Lost of lock interrupt for SUB_INC1	GTM_DPLL_IRQ[14]
737	1B84h	DPLL_EI: DPLL Error interrupt	GTM_DPLL_IRQ[15]
738	1B88h	DPLL_GL2I: DPLL Get of lock interrupt for SUB_INC2	GTM_DPLL_IRQ[16]
739	1B8Ch	DPLL_LL2I: DPLL Lost of lock interrupt for SUB_INC2	GTM_DPLL_IRQ[17]
740	1B90h	DPLL_TE0I: DPLL TRIGGER event interrupt 0	GTM_DPLL_IRQ[18]
741	1B94h	DPLL_TE1I: DPLL TRIGGER event interrupt 1	GTM_DPLL_IRQ[19]
742	1B98h	DPLL_TE2I: DPLL TRIGGER event interrupt 2	GTM_DPLL_IRQ[20]
743	1B9Ch	DPLL_TE3I: DPLL TRIGGER event interrupt 3	GTM_DPLL_IRQ[21]
744	1BA0h	DPLL_TE4I; DPLL TRIGGER event interrupt 4	GTM_DPLL_IRQ[22]
745	1BA4h	GTM DPLL trigger calculated duration interrupt	GTM_ICM_IRQG_1[DPLL_CDIT_IRQ]
746	1BA8h	GTM DPLL state calculated duration interrupt	GTM_ICM_IRQG_1[DPLL_CDIS_IRQ]
747	1BACh	GTM DPLL trigger out of range interrupt	GTM_ICM_IRQG_1[DPLL_TORI_IRQ]
748	1BB0h	GTM DPLL status out of range interrupt	GTM_ICM_IRQG_1[DPLL_SORI_IRQ]
749	1BB4h	GTM TIM0 Channel 0 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH0_IRQ]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
750	1BB8h	GTM TIM0 Channel 1 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH1_IRQ]
751	1BBCh	GTM TIM0 Channel 2 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH2_IRQ]
752	1BC0h	GTM TIM0 Channel 3 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH3_IRQ]
753	1BC4h	GTM TIM0 Channel 4 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH4_IRQ]
754	1BC8h	GTM TIM0 Channel 5 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH5_IRQ]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
755	1BCCh	GTM TIM0 Channel 6 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH6_IRQ]
756	1BD0h	GTM TIM0 Channel 7 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM0_CH7_IRQ]
757	1BD4h	GTM TIM1 Channel 0 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH0_IRQ]
758	1BD8h	GTM TIM1 Channel 1 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH1_IRQ]
759	1BDCh	GTM TIM1 Channel 2 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH2_IRQ]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
760	1BE0h	GTM TIM1 Channel 3 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) ECNT – Counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH3_IRQ]
761	1BE4h	GTM TIM1 Channel 4 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH4_IRQ]
762	1BE8h	GTM TIM1 Channel 5 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH5_IRQ]
763	1BECh	GTM TIM1 Channel 6 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH6_IRQ]
764	1BF0h	GTM TIM1 Channel 7 shared interrupt: – New measurement value detected by SMU for channel (NEWVALx_IRQ) – ECNT counter overflow of channel (ECNTOFLx_IRQ) – SMU counter overflow (CNTOFLx_IRQ) – GPRz data overflow (GPRzOFLx_IRQ) – Timeout of input signal (TOx_IRQ) – Glitch detected on input (GLITCHDETx_IRQ)	GTM_ICM_IRQG_2[TIM1_CH7_IRQ]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
781	1C34h	GTM MCS0 Channel 0 interrupt	GTM_ICM_IRQG_4[MCS0_CH0_IRQ]
782	1C38h	GTM MCS0 Channel 1 interrupt	GTM_ICM_IRQG_4[MCS0_CH1_IRQ]
783	1C3Ch	GTM MCS0 Channel 2 interrupt	GTM_ICM_IRQG_4[MCS0_CH2_IRQ]
784	1C40h	GTM MCS0 Channel 3 interrupt	GTM_ICM_IRQG_4[MCS0_CH3_IRQ]
785	1C44h	GTM MCS0 Channel 4 interrupt	GTM_ICM_IRQG_4[MCS0_CH4_IRQ]
786	1C48h	GTM MCS0 Channel 5 interrupt	GTM_ICM_IRQG_4[MCS0_CH5_IRQ]
787	1C4Ch	GTM MCS0 Channel 6 interrupt	GTM_ICM_IRQG_4[MCS0_CH6_IRQ]
788	1C50h	GTM MCS0 Channel 7 interrupt	GTM_ICM_IRQG_4[MCS0_CH7_IRQ]
789	1C54h	GTM MCS1 Channel 0 interrupt	GTM_ICM_IRQG_4[MCS1_CH0_IRQ]
790	1C58h	GTM MCS1 Channel 1 interrupt	GTM_ICM_IRQG_4[MCS1_CH1_IRQ]
791	1C5Ch	GTM MCS1 Channel 2 interrupt	GTM_ICM_IRQG_4[MCS1_CH2_IRQ]
792	1C60h	GTM MCS1 Channel 3 interrupt	GTM_ICM_IRQG_4[MCS1_CH3_IRQ]
793	1C64h	GTM MCS1 Channel 4 interrupt	GTM_ICM_IRQG_4[MCS1_CH4_IRQ]
794	1C68h	GTM MCS1 Channel 5 interrupt	GTM_ICM_IRQG_4[MCS1_CH5_IRQ]
795	1C6Ch	GTM MCS1 Channel 6 interrupt	GTM_ICM_IRQG_4[MCS1_CH6_IRQ]
796	1C70h	GTM MCS1 Channel 7 interrupt	GTM_ICM_IRQG_4[MCS1_CH7_IRQ]
813	1CB4h	GTM TOM0 Event 0 combined interrupt: – TOM0 channel 0 event – TOM0 channel 1 event	GTM_ICM_IRQG_9[TOM0_CH0_IRQ] | GTM_ICM_IRQG_9[TOM0_CH1_IRQ]
814	1CB8h	GTM TOM0 Event 1 combined interrupt: – TOM0 channel 2 event – TOM0 channel 3 event	GTM_ICM_IRQG_9[TOM0_CH2_IRQ] | GTM_ICM_IRQG_9[TOM0_CH3_IRQ]
815	1CBCh	GTM TOM0 Event 2 combined interrupt: – TOM0 channel 4 event – TOM0 channel 5 event	GTM_ICM_IRQG_9[TOM0_CH4_IRQ] | GTM_ICM_IRQG_9[TOM0_CH5_IRQ]
816	1CC0h	GTM TOM0 Event 3 combined interrupt: – TOM0 channel 6 event – TOM0 channel 7 event	GTM_ICM_IRQG_9[TOM0_CH6_IRQ] | GTM_ICM_IRQG_9[TOM0_CH7_IRQ]
817	1CC4h	GTM TOM0 Event 4 combined interrupt: – ATOM1 channel 8 event – ATOM1 channel 9 event	GTM_ICM_IRQG_9[TOM0_CH8_IRQ] | GTM_ICM_IRQG_9[TOM0_CH9_IRQ]
818	1CC8h	GTM TOM0 Event 5 combined interrupt: – TOM0 channel 10 event – TOM0 channel 11 event	GTM_ICM_IRQG_9[TOM0_CH10_IRQ] | GTM_ICM_IRQG_9[TOM0_CH11_IRQ]
819	1CCCh	GTM TOM0 Event 6 combined interrupt: – TOM0 channel 12 event – TOM0 channel 13 event	GTM_ICM_IRQG_9[TOM0_CH12_IRQ] | GTM_ICM_IRQG_9[TOM0_CH13_IRQ]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
820	1CD0h	GTM TOM0 Event 7 combined interrupt: – TOM0 channel 14 event – TOM0 channel 15 event	GTM_ICM_IRQG_9[TOM0_CH14_IRQ] | GTM_ICM_IRQG_9[TOM0_CH15_IRQ]
821	1CD4h	GTM TOM1 Event 0 combined interrupt: – TOM1 channel 0 event – TOM1 channel 1 event	GTM_ICM_IRQG_9[TOM1_CH0_IRQ] | GTM_ICM_IRQG_9[TOM1_CH1_IRQ]
822	1CD8h	GTM TOM1 Event 1 combined interrupt: – TOM1 channel 2 event – TOM1 channel 3 event	GTM_ICM_IRQG_9[TOM1_CH2_IRQ] | GTM_ICM_IRQG_9[TOM1_CH3_IRQ]
823	1CDCh	GTM TOM1 Event 2 combined interrupt: – TOM1 channel 4 event – TOM1 channel 5 event	GTM_ICM_IRQG_9[TOM1_CH4_IRQ] | GTM_ICM_IRQG_9[TOM1_CH5_IRQ]
824	1CE0h	GTM TOM1 Event 3 combined interrupt: – TOM1 channel 6 event – TOM1 channel 7 event	GTM_ICM_IRQG_9[TOM1_CH6_IRQ] | GTM_ICM_IRQG_9[TOM1_CH7_IRQ]
825	1CE4h	GTM TOM1 Event 4 combined interrupt: – ATOM1 channel 8 event – ATOM1 channel 9 event	GTM_ICM_IRQG_9[TOM1_CH8_IRQ] | GTM_ICM_IRQG_9[TOM1_CH9_IRQ]
826	1CE8h	GTM TOM1 Event 5 combined interrupt: – TOM1 channel 10 event – TOM1 channel 11 event	GTM_ICM_IRQG_9[TOM1_CH10_IRQ] | GTM_ICM_IRQG_9[TOM1_CH11_IRQ]
827	1CECh	GTM TOM1 Event 6 combined interrupt: – TOM1 channel 12 event – TOM1 channel 13 event	GTM_ICM_IRQG_9[TOM1_CH12_IRQ] | GTM_ICM_IRQG_9[TOM1_CH13_IRQ]
828	1CF0h	GTM TOM1 Event 7 combined interrupt: – TOM1 channel 14 event – TOM1 channel 15 event	GTM_ICM_IRQG_9[TOM1_CH14_IRQ] | GTM_ICM_IRQG_9[TOM1_CH15_IRQ]
837		GTM_ATOM0_IRQ[0]	gtm_icm.gtm_atom0_irq[0]
838		GTM_ATOM0_IRQ[1]	gtm_icm.gtm_atom0_irq[1]
839		GTM_ATOM0_IRQ[2]	gtm_icm.gtm_atom0_irq[2]
840		GTM_ATOM0_IRQ[3]	gtm_icm.gtm_atom0_irq[3]
841		GTM_ATOM1_IRQ[0]	gtm_icm.gtm_atom1_irq[0]
842		GTM_ATOM1_IRQ[1]	gtm_icm.gtm_atom1_irq[1]
843		GTM_ATOM1_IRQ[2]	gtm_icm.gtm_atom1_irq[2]
844		GTM_ATOM1_IRQ[3]	gtm_icm.gtm_atom1_irq[3]
845		GTM_ATOM2_IRQ[0]	gtm_icm.gtm_atom2_irq[0]
846		GTM_ATOM2_IRQ[1]	gtm_icm.gtm_atom2_irq[1]
847		GTM_ATOM2_IRQ[2]	gtm_icm.gtm_atom2_irq[2]
848		GTM_ATOM2_IRQ[3]	gtm_icm.gtm_atom2_irq[3]

Table 22. Interrupt sources(Continued)

IRQ #	Offset	Source description	Source name
931		GTM_ERR_IRQ	gtm_err_irq
932	1E90	Decimation filter input buffer interrupt request flag	DECI_FILTR[IBIF]
933	1E94	Decimation filter output buffer interrupt request flag	DECI_FILTR[OBIF]
934	1E98	Decimation filter error interrupt request flag	DECI_FILTR[DIVR] DECI_FILTR[OVF] DECI_FILTR[OVR] DECI_FILTR[IVR] DECI_FILTR[SVR] DECI_FILTR[SCE] DECI_FILTR[SSE]
935	1E9C	Clock Calibration on CAN event	cu_int

6.3.6.3 Interrupt latency

The interrupt system 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.

The latency time is divided into 3 general categories, as per the following table:

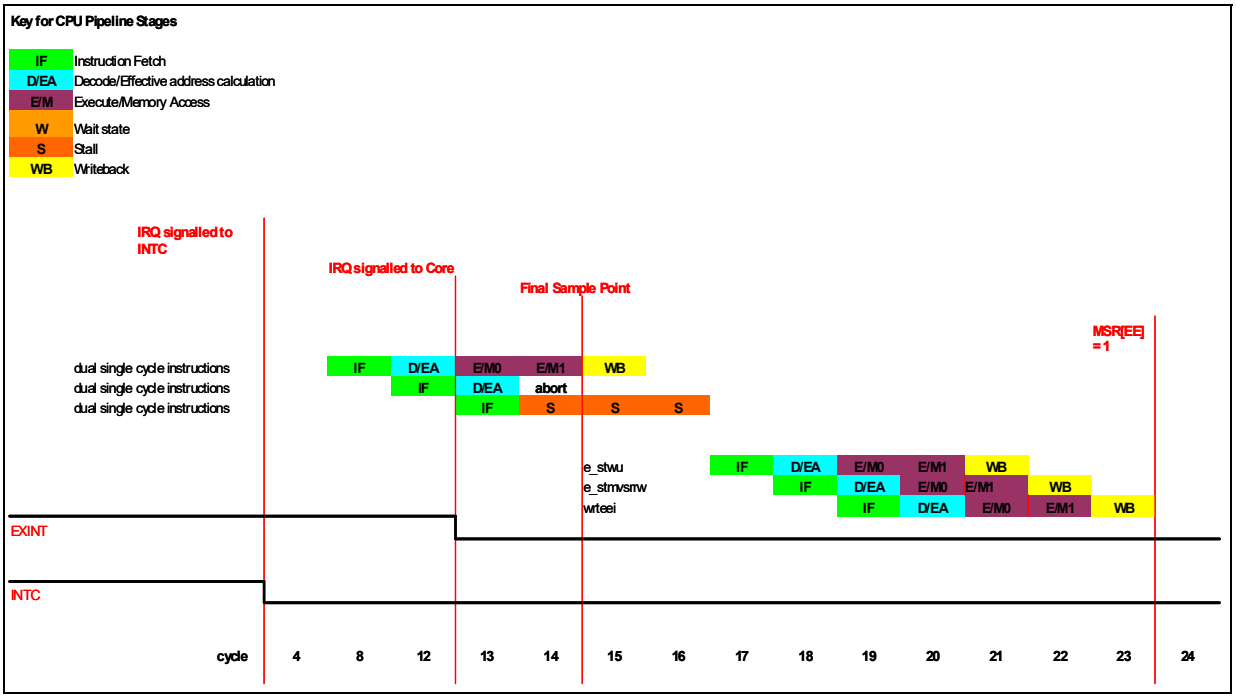
Table 23. Latency time categories

Category	Details
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 3 instructions needed to save essential registers and enable interrupts

INTC operates at 40 MHz and the CPU operates at 80 MHz for SPC572Lx, so the interrupt latency is:

- INTC latency = 75 ns
- CPU latency = 50 ns
- Software latency = 87.5 ns
- Interrupt latency total = 212.5 ns

Figure 10. Interrupt timing



6.3.7 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 11. DMA controller configuration

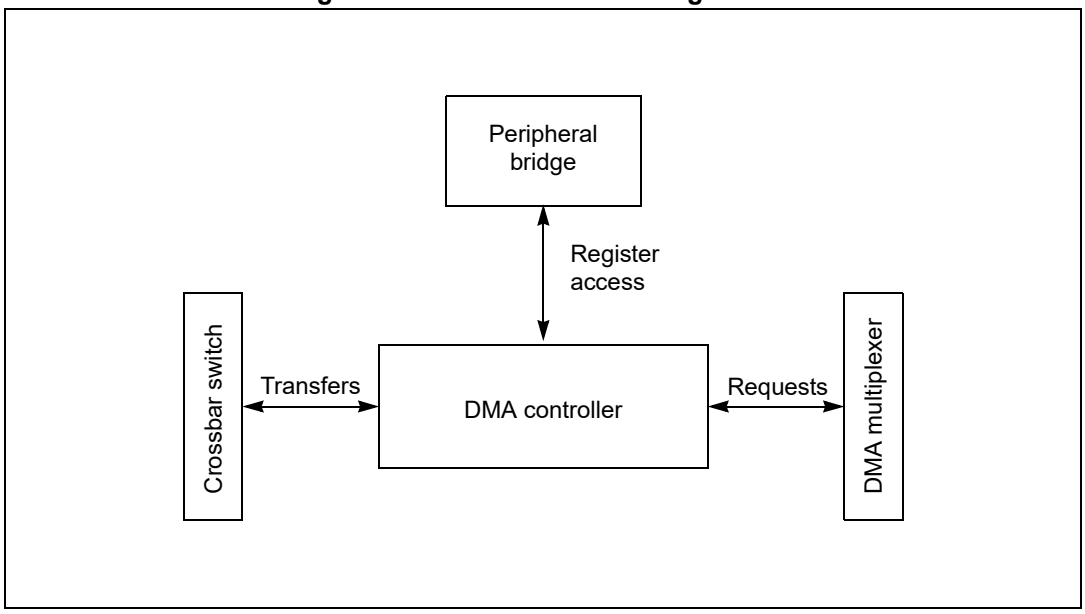


Table 24. Reference links to related information

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 15: Peripheral Bridge*
Clocking	—	*Chapter 21: Clocking*
Transfers	Crossbar switch	*Chapter 14: Crossbar Switch (XBAR)*

6.3.8 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.

6.3.8.1 DMA channel assignment

SPC572Lx has two multiplexers. *Table 25* lists which channels apply to each multiplexer.

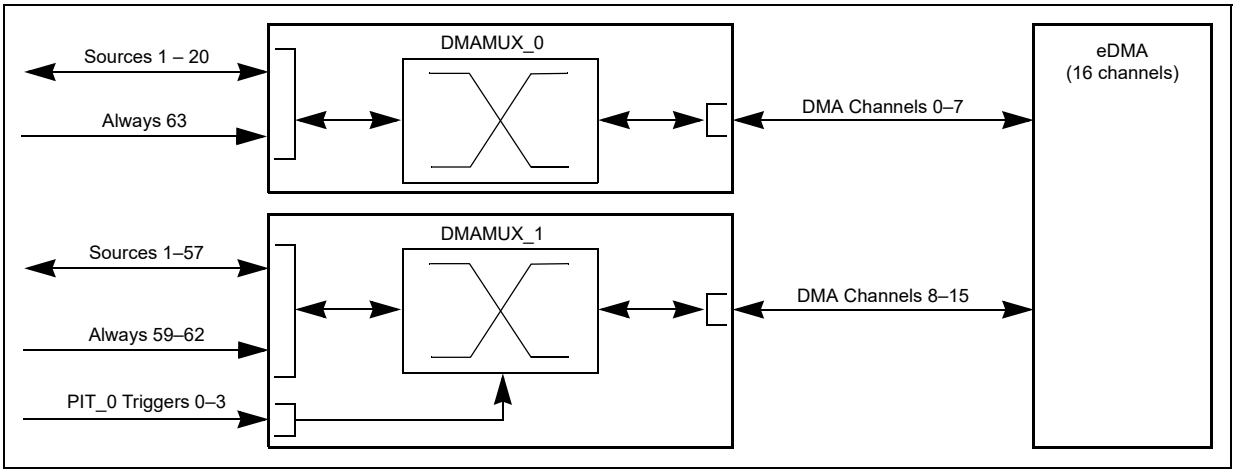
Table 25. SPC572Lx multiplexer and channel list

DMAMUX instance	DMA channel
0	0–7
1	8–15

6.3.8.2 Block diagram

The connections of the DMA channel multiplexers to the eDMA are shown in *Figure 12*. One DMAMUX is instantiated for every eight DMA channels.

Figure 12. DMA channel multiplexing



6.3.8.3 DMA request mapping

The mapping of the peripheral DMA requests to the DMA_CH_MUX input sources is shown in *Table 26*.

Table 26. DMAMUX peripheral DMA request to input source mapping

Source	Peripheral DMA requests	
	DMAMUX_0	DMAMUX_1
0	Reserved	Reserved
1	ADC_SAR_0 EOC	Reserved
2	Reserved	Reserved
3	ADC_SAR_B EOC	LINFlex_0 RX
4	Reserved	LINFlex_0 TX
5	DSPI_0 RX	LINFlex_1 RX
6	DSPI_0 TX	LINFlex_1 TX
7	DSPI_4 RX	LINFlex_14 RX
8	DSPI_4 Tx	LINFlex_14 TX
9	Reserved	SENT_0 RX FAST
10	ADC_SAR_4 EOC	SENT_0 RX SLOW
11	ADC_SD_3 EOC	SIPI CH0
12	MCAN_1	SIPI CH1
13	MCAN_2	SIPI CH2
14	SENT_0 RX FAST	SIPI CH3
15	SENT_0 RX SLOW	SIUL2 REQ0
16	LINFlex_0 RX	SIUL2 REQ1
17	LINFlex_0 TX	GTM_TIM0_IRQ0
18	LINFlex_14 RX	GTM_TIM0_IRQ1
19	DSPI_0 CMD	GTM_TIM0_IRQ2
20	DSPI_4 CMD	GTM_TIM0_IRQ3
21	Reserved	GTM_TIM0_IRQ4
22	Reserved	GTM_TIM0_IRQ5
23	Reserved	GTM_TIM0_IRQ6
24	Reserved	GTM_TIM0_IRQ7
25	Reserved	GTM_TOM0_IRQ0
26	Reserved	GTM_TOM0_IRQ1
27	Reserved	GTM_TOM0_IRQ2
28	Reserved	GTM_TOM0_IRQ3
29	Reserved	GTM_TOM0_IRQ4

Table 26. DMAMUX peripheral DMA request to input source mapping(Continued)

Source	Peripheral DMA requests	
	DMAMUX_0	DMAMUX_1
30	Reserved	GTM_TOM0_IRQ5
31	Reserved	GTM_TOM0_IRQ6
32	Reserved	GTM_TOM0_IRQ7
33	Reserved	GTM_ATOM0_IRQ0
34	Reserved	GTM_ATOM0_IRQ1
35	Reserved	GTM_ATOM0_IRQ2
36	Reserved	GTM_ATOM0_IRQ3
37	Reserved	GTM_MCS0_IRQ0
38	Reserved	GTM_MCS0_IRQ1
39	Reserved	GTM_MCS0_IRQ2
40	Reserved	GTM_MCS0_IRQ3
41	Reserved	GTM_MCS0_IRQ4
42	Reserved	GTM_MCS0_IRQ5
43	Reserved	GTM_MCS0_IRQ6
44	Reserved	GTM_MCS0_IRQ7
45	Reserved	Reserved
46	Reserved	Reserved
47	Reserved	Reserved
48	Reserved	Reserved
49	Reserved	Reserved
50	Reserved	Reserved
51	Reserved	DECFIL_FILL_BUF
52	Reserved	DECFIL_DRAIN_BUF
53	Reserved	DSPI_0 RX
54	Reserved	DSPI_0 TX
55	Reserved	ADC_SAR_0 EOC
56	Reserved	ADC_SAR_4 EOC
57	Reserved	ADC_SD_3 EOC
58	Reserved	Reserved
59	Reserved	Always on
60	Reserved	Always on
61	Reserved	Always on

Table 26. DMAMUX peripheral DMA request to input source mapping(Continued)

Source	Peripheral DMA requests	
	DMAMUX_0	DMAMUX_1
62	Reserved	Always on
63	Always on	Reserved

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 27*.

Table 27. PIT_0 periodic trigger to DMA channel mapping

PIT_0 periodic trigger	DMA channel
0	8
1	9
2	10
3	11

6.3.8.4 DMAMUX memory map

The memory maps for all DMA_CH_MUX instances is shown in *Table 28*. Each instance is assigned to its own peripheral bridge slot. The base address of each instance is determined by its peripheral bridge slot assignment.

Table 28. DMAMUX memory map

Offset(1)	Register(2)	Access
0x0000	DMAMUX_0 CHCFG0–CHCFG7—Channel configuration registers	R/W
0x0200	DMAMUX_1 CHCFG0–CHCFG7—Channel configuration registers	R/W

1. Offsets are from DMAMUX_0 base address, shown in *Chapter 6: Device configuration*
2. Each channel must be configured separately via settings in a dedicated configuration channel. There is one configuration register per channel for each DMAMUX. See the DMAMUX chapter for details.

6.4 Clocking

See *Chapter 21: 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).

6.5 Memories and memory interfaces

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

Table 29. Reference links to related information

Topic	Reference
Embedded memories	*Chapter 3: Embedded memories*
System memory map	*Chapter 5: Memory map*
Platform Flash	*Chapter 28: Flash memory controller (PFLASH Controller)*
Platform RAM Controller	*Chapter 27: RAM controller (PRAM)*
Embedded flash memory	*Chapter 28: Flash memory controller (PFLASH Controller)*
Decorated Storage Memory Controller	*Chapter 30: Decorated Storage Memory Controller (DSMC)*

6.5.1 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 *Table 30*. Note that these are the same master assignments used elsewhere in the chip, e.g., the SMPU.

Table 30. Flash controller bus master assignments

Bus master number	Bus master
0	Core
1	Reserved
2	Reserved
3	DMA
4	Ethernet
5	Reserved
6	Reserved
7	SIPI/LFAST
8	Core Debug
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved

Table 30. Flash controller bus master assignments(Continued)

Bus master number	Bus master
14	Reserved
15	Reserved

Table 31. Reference links to related information

Topic	Related module	Reference
Full description	Flash Memory Controller (PFLASH)	*Chapter 28: Flash memory controller (PFLASH Controller)*
Flash memory architecture overview	Flash memory, Flash controller	*Section 3.3: Embedded flash memory*
Flash memory	Flash memory	*Chapter 29: Embedded Flash Memory (MP55)*
System memory map	—	*Chapter 5: Memory map*

6.5.2 Decorated Storage Memory Controller (DSMC)

The DSMC supports five store and three load operations, including

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

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

The module includes error checking logic that validates the decoration field is properly defined (all illegal commands are rejected and the transfer error terminated). Additionally, the decorated storage memory controller only operates on aligned, single data transfers (misalignment and/or bursts are not supported and error terminated if attempted).

The DSMC is instantiated multiple times within the core platform and physically resides between the slave ports of the crossbar and the targeted memory controllers.

For the SPC572Lx device, the DSMC is instantiated two times:

- System RAM
- AIPS-Lite 0 (Bridge to peripheral subsystem)

6.5.3 ECC Error Reporting Module

The Error Reporting Module (ERM) provides information and optional interrupt notification on memory error events associated with ECC (Error Correction Code).

The ERM collects ECC events on memory accesses for local memory arrays, such as flash, system RAM or peripheral RAMs.

Table 32 lists the memory assignment to the ERM channels on SPC572Lx.

Table 32. ERM memory assignment

ERM memory channel	Assigned memory
Memory0	System SRAM (read by crossbar slave port S1)
Memory1	eDMA RAM
Memory2	Fast Ethernet Controller MIB RAM
Memory3	Fast Ethernet Controller FIFO RAM
Memory4	FLASH (code flash, data flash, flash controller RAM) Overlay RAM (calibration RAM)
Memory5	System SRAM (read by pFlash controller through overlay backdoor)
Memory6	M_CAN shared RAM
Memory7	GTM RAM

6.5.4 ECC Error Injection Module

The Error Injection Module (EIM) provides support for inducing single-bit and multi-bit inversions on read data when accessing peripheral RAM.

Table 33 lists the memory assignment to the EIM channels on SPC572Lx.

Table 33. EIM memory assignment

EIM channel	Assigned memory
Memory0	System SRAM
Memory1	eDMA RAM
Memory2	Fast Ethernet Controller MIB RAM
Memory3	Fast Ethernet Controller FIFO RAM
Memory4	Overlay RAM0
Memory5	Overlay RAM1

The Error Injection on the Flash memory can be handled by reading the ECC/EDC test pattern available in the UTEST memory.

6.6 Analog modules

This section highlights the following analog modules that are implemented on SPC572Lx:

Table 34. Analog modules

Module or Subsystem	Device-specific information	Block details
Temperature sensor	*Section 6.6.1: Temperature sensor configuration*	*Chapter 38: Temperature Sensor*
ADC subsystem	*Section 34.1: ADC overview*	See individual modules
Sigma-Delta Analog-to-Digital Converters (SDADCs)	*Section 34.2.1: Sigma-Delta Analog-to-Digital Converter (SDADC)*	*Chapter 35: Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface*
Successive Approximation Register Analog-to-Digital Converters (SARADCs)	*Section 34.2.2: Successive Approximation Register Analog-to-Digital Converter (SARADC)*	*Chapter 36: Successive Approximation Register Analog-to-Digital Converter (SARADC)*

6.6.1 Temperature sensor configuration

The temperature sensor is used to measure the temperature of the SPC572Lx device. In general, the temperature sensor generates an analog voltage that is linearly proportional to temperature.

During production testing of the device, the output of the temperature sensor is sampled by SARB input channel 120 of the onboard ADC module at a predefined high temperature and also at a predefined low temperature. These two temperature sensor readings are used to calculate two constants, TSCA and TSCB, according to the formulas in the *Chapter 38: Temperature Sensor*.

Once calculated, the values for TSCA and TSCB are programmed during production test into the UTEST Flash Memory.

The temperature sensor bandgap voltage used in the temperature calculation is sampled by SARB input channel 121 of the onboard ADC module.

Software must perform the following steps in order to calculate device temperature:

1. Measure the bandgap voltage, $V_{BG}(T)$, using SARB input channel 121 of the onboard ADC module.
2. Calculate $V_{BG_CODE}(T)$ according to the formula in *Chapter 38: Temperature Sensor*.
3. Measure the temperature sensor output voltage, $V_{TSENS}(T)$, using SARB input channel 120 of the onboard ADC module.
4. Calculate $T_{TSENS_CODE}(T)$ according to the formula in *Chapter 38: Temperature Sensor*.
5. Retrieve parameters TSCA and TSCB from flash memory.
6. Calculate temperature (T) according to the formula in *Chapter 38: Temperature Sensor*.

6.6.2 Analog-to-Digital Converters (ADC) configuration

See *Chapter 34: Analog-to-digital converter (ADC) configuration* for an overview of the ADC subsystem, including a block diagram and a summary of analog input pin multiplexing. The chapter also provides configuration details for the SARADCs and SDADCs.

6.6.2.1 SDADC configuration

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

SPC572Lx includes one independent 16-bit SDADC.

For details, see *Chapter 34: Analog-to-digital converter (ADC) configuration*.

6.6.2.2 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.

SPC572Lx includes three independent 12-bit SARADCs:

- SARADC_B
- SARADC_0
- SARADC_4

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

For details, see *Chapter 34: Analog-to-digital converter (ADC) configuration*.

6.7 Timers

Four types of timers and a complex timer subsystem are implemented on SPC572Lx.

Table 35. Timers

Timer	Device-specific information	Block details
System Timer Module (STM)	*Section 6.7.1: System Timer Module (STM) configuration*	*Chapter 39: System Timer Module (STM)*
Software Watchdog Timer (SWT)	*Section 6.7.2: Software Watchdog Timer (SWT) configuration*	*Chapter 40: Software Watchdog Timer (SWT)*
Periodic Interrupt Timer (PIT)	*Section 6.7.3: PIT configuration*	*Chapter 41: Periodic Interrupt Timer (PIT)*
GTM101 Generic Timer Module	*Section 6.7.4: GTM integration module configuration*	*Chapter 43: GTM101 Integration (GTMINT) Module*

The core has a dedicated System Timer Module (STM) and a dedicated Software Watchdog Timer (SWT).

Two Periodic Interrupt Timer (PIT) module instances are implemented. One PIT module is for 64-bit timer implementation by way of two 32-bit timer channels (0 and 1), with chaining mode. The timer value of this 64-bit timer is unchanged by a non-destructive reset. The other PIT module is for general use and has four 32-bit timer channels with no chaining mode.

The GTM101 is an implementation of the Robert Bosch GmbH GTM timer subsystem. It is not a single module, but a complex timer subsystem that consists of many modules that can be used to implement highly complex timer functions.

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

SPC572Lx includes one STM (STM_2)—which runs on the PBRIDGE_A clock domain—with four 32-bit compare channels and fourteen 32-bit registers.

6.7.2 Software Watchdog Timer (SWT) configuration

SPC572Lx includes two SWTs.

Table 36. Software Watchdog Timer (SWT) instances

Instance	Description
SWT2	Intended for Peripheral Core_2 in the peripheral clock domain.
SWT3	Intended as security watchdog timer in the peripheral clock domain. Elapsed times are recorded in timeout register.

The SWT has the following features:

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

6.7.2.1 Security watchdog

SWT3 (SWT3 Security) is identical to the other SWT modules, except the Instruction Address Compare (IAC) feature is not instantiated. The CPU can use an SWT to ensure software tasks continue to execute correctly.

If a CPU malfunctions (either software or hardware) the SWT is no longer serviced and a reset occurs. This is referred to as a Task Watchdog. The Security Watchdog is slightly different to the Task Watchdog.

The Security watchdog is intended to be serviced when the CPU executes an instruction at a specific address location within some fundamental section of security code. If this address location is not executed regularly, it implies that a malicious security attack has occurred and the device should be reset.

This security code can be executed on the CPU. The SWT for the CPU performs the Security Watchdog (SWT2), because it includes the IAC feature. The Task Watchdog for that CPU needs to use SWT3 (Security), which does not include the IAC feature. As a

result, the SWT3 (Security) is never used to perform the Security Watchdog; it is employed for the Task Watchdog in place of the SWT that is being used for the Security Watchdog.

Additionally, the Security watchdog is deactivated by a device power-on-reset.

6.7.2.2 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.

6.7.2.3 Default configuration

On SPC572Lx, the SWTs come out of reset with the default reset values summarized in *Table 37*. Minimum timeout value is 256 cycles for SWT2 and SWT3.

Table 37. SWT register reset values

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

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

6.7.2.4 Service mode watchpoint input

Each SWT comes out of reset in 'Fixed Service Sequence' mode (SWT_CR[SMD] = 2'b00), but the Instruction Address Compare 8 (IAC8) register (SPR 568) can be used for servicing by using the SWT jd_watchpt input.

To use SWT address execution modes for this purpose, the e200z2 and e200z4 core IAC8 register signal jd_watchpt[15] is connected to the SWT jd_watchpt input. (See *Section 57.6: Watchpoint support* for details.) The jd_watchpt[15] signal tells the SWT that an IAC8 compare occurred.

Set SWT_CR[SMD] to one of the following so the SWT is serviced by executing code at the address loaded into the IAC8 register:

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

6.7.3 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 13. PIT configuration

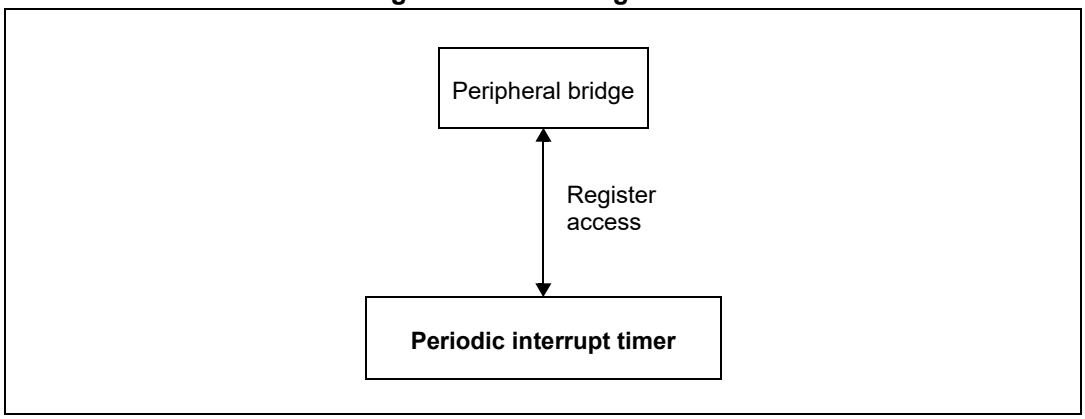


Table 38. Reference links to related information

Topic	Related module	Reference
Full description	PIT	*Chapter 41: Periodic Interrupt Timer (PIT)*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 21: Clocking*
Power management	—	*Chapter 9: Power management*
DMA channel assignments	Direct memory access multiplexer	*Chapter 20: Direct Memory Access Multiplexer (DMAMUX)*

6.7.3.1 PIT instantiation

Table 39. PIT information

PIT module	Number of channels	Description
PIT0	4	PIT0 has 4 standard channels. In PIT0, all registers are cleared by any reset.
PIT1	2	PIT1 has chaining mode to implement a 64-bit timer and the timer value is unchanged by a functional reset.

6.7.3.2 PIT/DMA periodic trigger assignments

In PIT1, DMA is not supported. PIT0 generates periodic trigger events to the DMA Mux. To check the mapping of the PIT0 channel number with the corresponding DMA channel number, refer to the *Table 26: DMAMUX peripheral DMA request to input source mapping*.

6.7.3.3 PIT registers

Table 40 and *Table 41* show the implementation of PIT registers in this MCU.

Table 40. PIT1 memory map

Absolute address (hex)	Register name
FFF8_0000	PIT Module Control Register (PIT1_MCR)(1)
FFF8_00E0 – FFF8_00E4	Reserved
FFF8_00F0 – FFF8_00FC	Reserved
FFF8_0100 – FFF8_017C	Timer Channel [0:3] Registers (PIT1_LDVALn, PIT1_CVALn, PIT1_TCTRLn, and PIT1_TFLGn)
FFF8_0180 – FFF8_3FFF	Reserved

1. The reset value of PIT1_MCR is 0x02.

Table 41. PIT0 memory map

Absolute address (hex)	Register name
FFF8_4000	PIT Module Control Register (PIT0_MCR)(1)
FFF8_40E0	PIT Upper Lifetime Timer Register (PIT0_LTMR64H)
FFF8_40E4	PIT Lower Lifetime Timer Register (PIT0_LTMR64L)
FFF8_40F0 – FFF8_40F8	Reserved
FFF8_4100 – FFF8_411C	Timer Channel [0:1] Registers (PIT0_LDVALn, PIT0_CVALn, PIT0_TCTRLn, and PIT0_TFLGn)
FFF8_4120 – FFF8_7FFF	Reserved

1. The reset value of PIT0_MCR is 0x06.

6.7.3.4 PIT clocking

The PIT0 standard channels are clocked with the peripheral clock (PER_CLK). The PIT 1 channels are clocked with the peripheral clock (PER_CLK).

6.7.4 GTM integration module configuration

The GTM101 integration module (GTMINT) is an interface to the General Timer Module 101 that adds ECC functionality and a Nexus-based system debugger interface (GTMDI). It has two control registers and provides access to the GTM101's memory-mapped registers. The GTM101 is detailed separately in the *GTM-IP_101_Specifcation*. See *Chapter 43: GTM101 Integration (GTMINT) Module* for more information.

6.8 Communication interfaces

6.8.1 FEC interface

The Fast Ethernet Controller on the SPC572Lx device only supports the Reduced Media Independent Interface (RMII) mode. The MII_MODE field in the Receive Control Register *Figure 710: Receive Control Register (RCR)* must be set by the user to '1' prior to FEC initialization — see *Section 48.5.7: User initialization (after setting ECR[ETHER_EN])*.

The MII Management Frame Register (MMFR) is not supported on SPC572Lx.

Note: The RMII_MODE field is fixed by hardware to '1', and is not configurable by the user.

Refer to *Chapter 48: Fast Ethernet Controller (FEC)* for detailed description.

6.8.2 CAN nodes

Table 42 lists the number of CAN nodes.

Table 42. CAN bus controllers

Type	Instances	DEBUG on CAN support	Clock calibration(1)
M_CAN	M_CAN_1	Yes	Yes
	M_CAN_2	Yes	Yes

1. Clock calibration based on M_CAN_1 messages

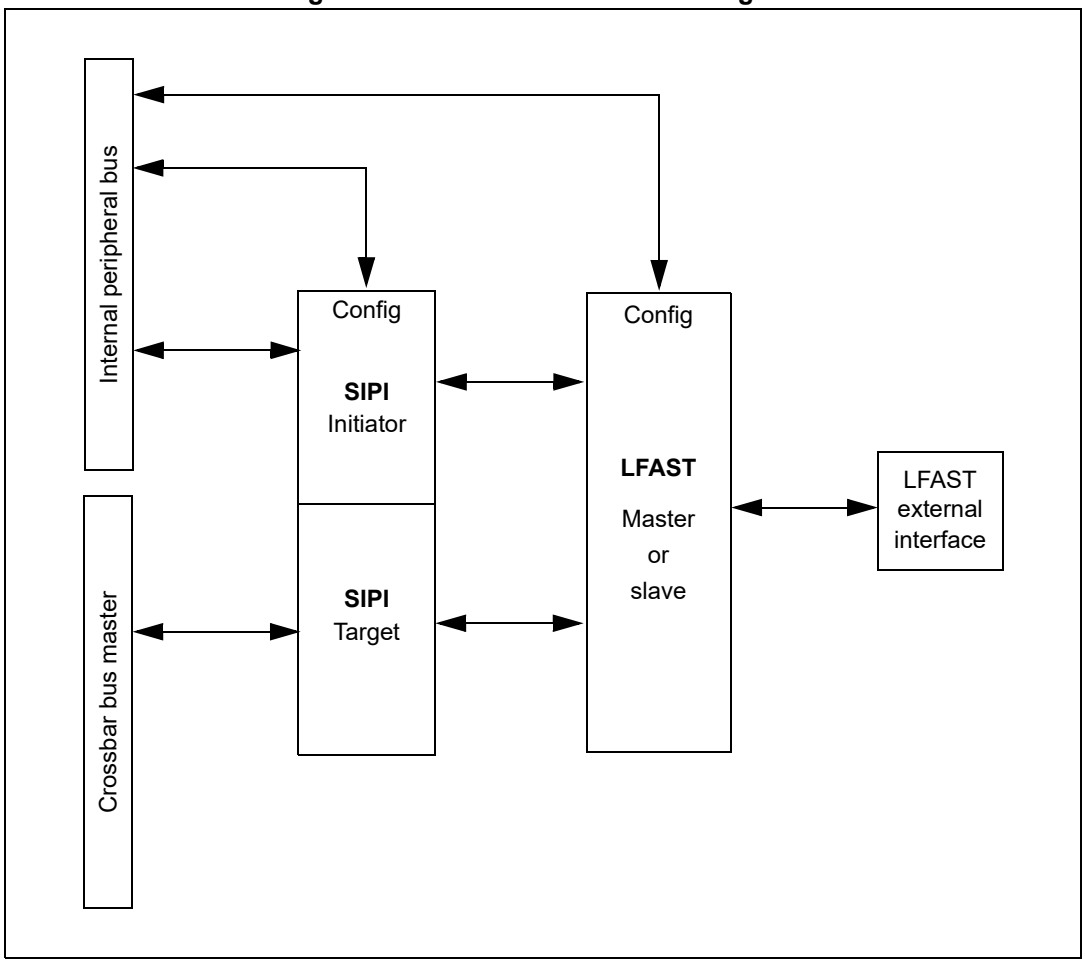
6.8.3 LFAST and SIPI

This section summarizes the module configuration in the controller. For a comprehensive description of SIPI and LFAST, reference the specific dedicated chapters for SIPI and LFAST.

6.8.3.1 Instantiation

The SIPI and LFAST are connected together appearing as a single unit. The LFAST portion of the two modules allows for high speed inter-device communications. The SIPI allows memory to be shared between devices which have SIPI and LFAST communication modules (see *Figure 14*).

Figure 14. LFAST and SIPI block diagram



6.8.3.2 Device specific features

The LFAST can operate in either slave or master mode configurations. The node running in master mode controls the serial link but SIPI in both master and slave nodes can act as both Initiator and Target for SIPI commands simultaneously. See the SIPI and LFAST dedicated chapters for detailed information on module operations. The SIPI CHIP ID = 0AF02041h for the SPC572Lx. SPC572Lx does not contain a dedicated LFAST PLL, but it uses the PLL_0 and the LFAST is used only as interprocessor connection (SIPI), but not for Debug or Test purposes.

6.8.3.3 Memory map and register description

Table 43. Reference text to related information

Topic	Related Module	Reference
Full Description	LFAST	*Chapter 47: LVDS Fast Asynchronous Serial Transmission – Interprocessor Communications*
Full Description	SIPI	*Chapter 45: Serial Interprocessor Interface (SIPI)*

Table 43. Reference text to related information(Continued)

Topic	Related Module	Reference
Direct Memory Access	eDMA	*Chapter 19: Enhanced Direct Memory Access (eDMA)*
System Memory Map	—	*Chapter 5: Memory map*

6.8.4 DSPI 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. DSPI configuration

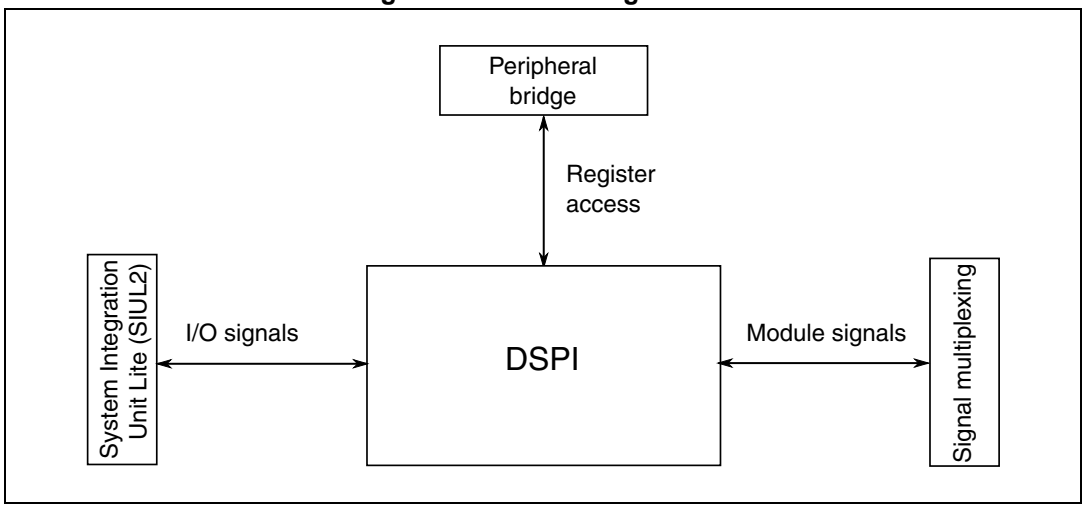


Table 44. Reference links to related information

Topic	Related module	Reference
Full description	DSPI	*Chapter 46: Deserial Serial Peripheral Interface (DSPI)*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 21: 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 multiplexer	*Chapter 20: Direct Memory Access Multiplexer (DMAMUX)*

6.8.4.1 DSPI instantiation

This device contains two DSPI modules. *Table 45* shows the instantiation information for each module.

Table 45. DSPI instantiation

DSPI module	CTARs	TX FIFO depth	RX FIFO depth	PCS signals	LVDS support	Data serialization support
DSPI0	8	4	4	PCS[7:0]	Not available	—
DSPI4	8	4	4	PCS[7:0]	Available(1)	64

1. Supports Microsecond Bus SCK and SOUT LVDS signal pairs.

6.8.4.2 Serial input for DSI mode

Serial input for the DSI mode is selected using the SIUL register(c):

- DSPI4 DSI0-31 is managed by MSCR656-687
- DSPI4 DSI32-63 is managed by MSCR936-967

6.8.4.3 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.

6.8.4.4 Serialized timed I/O

The timer sources and injection pin inputs to be DSPI serialized downstream outputs are selected through the SIUL2 Multiplexed Single Configuration Registers (MSCRs). Inversion of these signals is also handled in the SIUL2. Refer to the Signal Description chapter for details.

6.8.4.5 High speed and microsecond channel support

The baud rate generator in the DSPI is clocked from the peripheral clock. The DSPI module can operate at a baud rate up to half the peripheral clock. Peripheral clock frequency specification is 16 MHz to 120 MHz. As long as the peripheral clock is ≥ 100 MHz, 50 MHz baud is achievable.

To support microsecond channel functionality, two pairs of LVDS pads are implemented per DSPI. In the SIUL2_MSCR, the user can configure the desired port driving behavior (either CMOS or LDVS) for the DSPI SOUT and CLK signals for those instances specified with LVDS pins. Also configurable inside the SIUL2_MSCR are the input signals which should be serialized for TSB mode.

6.8.4.6 DSPI HT External Trigger

Use of external hardware triggers are not supported.

c. The SPC572Lx I/O Signal Description and Input Multiplexing Tables are contained in a Microsoft Excel® workbook file attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

6.8.4.7 DSPI ITSB Mode

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

6.8.4.8 DSPI registers

Table 46 shows the absolute addresses of DSPI module registers in this MCU.

Table 46. DSPI registers memory map

Absolute address (hex)	Module
FFE7_0000	DSPI0 registers
FFE7_8000	DSPI4 registers

6.8.4.8.1 DSPIx_MCR Implementation

Figure 16 shows the implementation of the DSPIx_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	0	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	1	0	0	1

Figure 16. DSPI Module Configuration Register (DSPI*x*_MCR)

Table 47. 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 47. DSPI*x*_MCR field descriptions(Continued)

Field	Description
2–3 DCONF	DSPI Configuration Selects among the different configurations of the DSPI. 00 SPI 01 DSI 10 CSI 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]/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.
8–15 PCSIS[7:0]	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx 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 PCSx is low. 1 The inactive state of PCSx is high.
16 Reserved	This read-only bitfield is reserved and always has the value zero.
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 default reset value of the MDIS bit is 1. When DSPI is used in Slave Mode, it is recommended to leave this bit set to ‘0’, since a slave does not have control over master transactions. 0 Enable DSPI clocks. 1 Allow external logic to disable DSPI clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the DSPI (both TXFIFO and CMD FIFO) operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 Tx FIFO is enabled. 1 Tx FIFO is disabled.

Table 47. DSPI*x*_MCR field descriptions(Continued)

Field	Description
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
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.

Table 47. DSPIx_MCR field descriptions(Continued)

Field	Description
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.

6.8.5 SENT Receiver (SRX) configuration

6.8.5.1 SRX channels

SPC572Lx has one instance of SRX (SRX_0) with four channels.

6.8.5.2 SRX registers

Table 48 lists the address offset and memory map for both SENT receiver (SRX) instances for this chip.

Table 48. SENT receiver memory map

Offset	Register name	SENT 0
0x0000	GBL_CTRL— Global Control Register	X
0x0004	CHNL_EN—Channel Enable Register	X
0x0008	GBL_STATUS—Global Status Register	X
0x000C	FMSG_RDY—Fast Message Ready Register	X
0x0010	SMSG_RDY—Slow Serial Message Ready Register	X
0x0014	Reserved	—
0x0018	DATA_CTRL1— Data Control Register 1	X
0x001C	Reserved	—
0x0020	Reserved	—
0x0024	Reserved	—
0x0028	FDMA_CTRL—Fast Message DMA Control Register	X
0x002C	SDMA_CTRL—Slow Serial Message DMA Control Register	X
0x0030	Reserved	—
0x0034	FRDY_IE—Fast Message Ready Interrupt Control Register	X
0x0038	SRDY_IE—Slow Serial Message Ready Interrupt Control Register	X
0x003C	Reserved	—
0x0040	DMA_FMSG_DATA—DMA Fast Message Read Register	X
0x0044	DMA_FMSG_CRC—DMA Fast Message CRC Read Register	X

Table 48. SENT receiver memory map(Continued)

Offset	Register name	SENT 0
0x0048	DMA_FMSG_TS—DMA Fast Message Time Stamp Read Register	X
0x004C	Reserved	—
0x0050	DMA_SMSG_BIT3—DMA Serial Message Read Register (Bit 3)	X
0x0054	DMA_SMSG_BIT2—DMA Serial Message Read Register (Bit 2)	X
0x0058	DMA_SMSG_TS—DMA Serial Message Time Stamp Read Register	X
0x005C	Reserved	—
0x0060	CH0_CLK_CTRL—Channel ‘n’ Clock Control Register	X
0x0064	CH0_STATUS—Channel ‘n’ Status Register	X
0x0068	CH0_CONFIG—Channel ‘n’ Configuration Register	X
0x006C	Reserved	—
0x0070	CH1_CLK_CTRL—Channel ‘n’ Clock Control Register	X
0x0074	CH1_STATUS—Channel ‘n’ Status Register	X
0x0078	CH1_CONFIG—Channel ‘n’ Configuration Register	X
0x007C	Reserved	—
0x0080	CH2_CLK_CTRL—Channel ‘n’ Clock Control Register	X
0x0084	CH2_STATUS—Channel ‘n’ Status Register	X
0x0088	CH2_CONFIG—Channel ‘n’ Configuration Register	X
0x008C	Reserved	—
0x0090	CH3_CLK_CTRL—Channel ‘n’ Clock Control Register	X
0x0094	CH3_STATUS—Channel ‘n’ Status Register	X
0x0098	CH3_CONFIG—Channel ‘n’ Configuration Register	X
0x009C – 0x015F	Reserved	—
0x0160	CH0_FMSG_DATA—Channel ‘n’ Fast Message Register	X
0x0164	CH0_FMSG_CRC—Channel ‘n’ Fast Message CRC Register	X
0x0168	CH0_FMSG_TS—Channel ‘n’ Fast Message Time Stamp Register	X
0x016C	CH0_SMSG_BIT3—Channel ‘n’ Serial Message Register (Bit 3)	X
0x0170	CH0_SMSG_BIT2—Channel ‘n’ Serial Message Register (Bit 2)	X
0x0174	CH0_SMSG_TS—Channel ‘n’ Serial Message Time Stamp Register	X
0x0178	CH1_FMSG_DATA—Channel ‘n’ Fast Message Register	X
0x017C	CH1_FMSG_CRC—Channel ‘n’ Fast Message CRC Register	X
0x0180	CH1_FMSG_TS—Channel ‘n’ Fast Message Time Stamp Register	X
0x0184	CH1_SMSG_BIT3—Channel ‘n’ Serial Message Register (Bit 3)	X
0x0188	CH1_SMSG_BIT2—Channel ‘n’ Serial Message Register (Bit 2)	X
0x018C	CH1_SMSG_TS—Channel ‘n’ Serial Message Time Stamp Register	X
0x0190	CH2_FMSG_DATA—Channel ‘n’ Fast Message Register	X

Table 48. SENT receiver memory map(Continued)

Offset	Register name	SENT 0
0x0194	CH2_FMSG_CRC—Channel ‘n’ Fast Message CRC Register	X
0x0198	CH2_FMSG_TS—Channel ‘n’ Fast Message Time Stamp Register	X
0x019C	CH2_SMSG_BIT3—Channel ‘n’ Serial Message Register (Bit 3)	X
0x01A0	CH2_SMSG_BIT2—Channel ‘n’ Serial Message Register (Bit 2)	X
0x01A4	CH2_SMSG_TS—Channel ‘n’ Serial Message Time Stamp Register	X
0x01A8	CH3_FMSG_DATA—Channel ‘n’ Fast Message Register	X
0x01AC	CH3_FMSG_CRC—Channel ‘n’ Fast Message CRC Register	X
0x01B0	CH3_FMSG_TS—Channel ‘n’ Fast Message Time Stamp Register	X
0x01B4	CH3_SMSG_BIT3—Channel ‘n’ Serial Message Register (Bit 3)	X
0x01B8	CH3_SMSG_BIT2—Channel ‘n’ Serial Message Register (Bit 2)	X
0x01BC	CH3_SMSG_TS—Channel ‘n’ Serial Message Time Stamp Register	X
0x01C0 – 0x02E0	Reserved	—

6.8.6 LINFlexD – Configurations

SPC572Lx has three instances of LINFlexD. *Table 49* lists the configuration for each instance.

Table 49. LINFlexD configurations

Description	linflex_0	linflex_1	linflex_14
Number of filters implemented	16	0	0
Number of Tx DMA channels	1	1	1
Number of Rx DMA channels	1	1	1
LIN operation mode	Master/Slave	Master	Master
Value of autosynchronization	1	0	0

6.8.6.1 LINFlexD implemented registers

Table 50 shows the registers implemented for each LINFlexD instance on the SPC572Lx chip.

Table 50. LINFlexD implemented registers

Register description	linflex_0 0xFFE8C000 Address offset (hex)	linflex_1 0xFFE90000 Address offset (hex)	linflex_14 0xFFEA8000 Address offset (hex)
LIN Control register 1 (LINCR1)	00	00	00
LIN Interrupt Enable register (LINIER)	04	04	04
LIN Status register (LINSR)	08	08	08

Table 50. LINFlexD implemented registers(Continued)

Register description	linflex_0 0xFFE8C000 Address offset (hex)	linflex_1 0xFFE90000 Address offset (hex)	linflex_14 0xFFEA8000 Address offset (hex)
LIN Error Status register (LINESR)	0C	0C	0C
UART Mode Control register (UARTCR)	10	10	10
UART Mode Status register (UARTSR)	14	14	14
LIN Timeout Control Status register (LINTCSR)	18	18	18
LIN Output Compare register (LINOCR)	1C	1C	1C
LIN Timeout Control register (LINTOCR)	20	20	20
LIN Fractional Baud Rate register (LINFBRR)	24	24	24
LIN Integer Baud Rate register (LINIBRR)	28	28	28
LIN Checksum Field register (LINCFR)	2C	2C	2C
LIN Control register 2 (LINCR2)	30	30	30
Buffer Identifier register (BIDR)	34	34	34
Buffer Data Register Least Significant (BDRL)	38	38	38
Buffer Data Register Most Significant (BDRM)	3C	3C	3C
Identifier Filter Enable register (IFER)	40	40	40
Identifier Filter Match index (IFMI)	44	44	44
Identifier Filter Mode register (IFMR)	48	48	48
Identifier Filter Control registers (IFCR0)	4C	—	
Identifier Filter Control registers (IFCR1)	50	—	
Identifier Filter Control registers (IFCR2)	54	—	
Identifier Filter Control registers (IFCR3)	58	—	
Identifier Filter Control registers (IFCR4)	5C	—	
Identifier Filter Control registers (IFCR5)	60	—	
Identifier Filter Control registers (IFCR6)	64	—	
Identifier Filter Control registers (IFCR7)	68	—	
Identifier Filter Control registers (IFCR8)	6C	—	
Identifier Filter Control registers (IFCR9)	70	—	
Identifier Filter Control registers (IFCR10)	74	—	
Identifier Filter Control registers (IFCR11)	78	—	
Identifier Filter Control registers (IFCR12)	7C	—	
Identifier Filter Control registers (IFCR13)	80	—	
Identifier Filter Control registers (IFCR14)	84	—	
Identifier Filter Control registers (IFCR15)	88	—	
Global Control register (GCR)	8C	4C	4C

Table 50. LINFlexD implemented registers(Continued)

Register description	linflex_0 0xFFE8C000 Address offset (hex)	linflex_1 0xFFE90000 Address offset (hex)	linflex_14 0xFFEA8000 Address offset (hex)
UART Preset Timeout register (UARTPTO)	90	50	50
UART Current Timeout register (UARTCTO)	94	54	54
DMA Tx Enable register (DMATXE)	98	58	58
DMA Rx Enable register (DMARXE)	9C	5C	5C

6.9 Reset and Boot modules

6.9.1 BAF 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.

6.9.1.1 Memory map

The BAF is located in a 16 KB block of flash that is mapped adjacent to the UTEST flash memory block. 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 51*.

Table 51. BAF memory organization

BAF Component	Address
BAF version number	0040_4000h
BAF code entry point	0040_4100h
BAF data area and stack	4000_0000h–4000_02FFh

6.9.1.2 Search boot header and boot options

The BAF searches the flash memory for the boot header, checking the first word location of the 4 × 256 KB blocks. The blocks are searched in the order shown in the following table and once a Boot Header is found, no further blocks are searched.

Table 52. Locations of boot headers

Search order	Block	Address
1st	256 KB Code flash memory block0	0100_0000h
2nd	256 KB Code flash memory block1	0104_0000h
3rd	256 KB Code flash memory block2	0108_0000h
4th	256 KB Code flash memory block3	010C_0000h

The first header found that contains the value 005Ah in the first half-word is valid for booting. The whole header structure is shown in *Table 53*.

Table 53. Boot header structure

Address offset	Contents
00h	Boot Header Configuration (see *Figure 17*)
04h	Boot Core Reset Vector
08h	Reserved
0Ch	Reserved
10h	Reserved
14h	Reserved

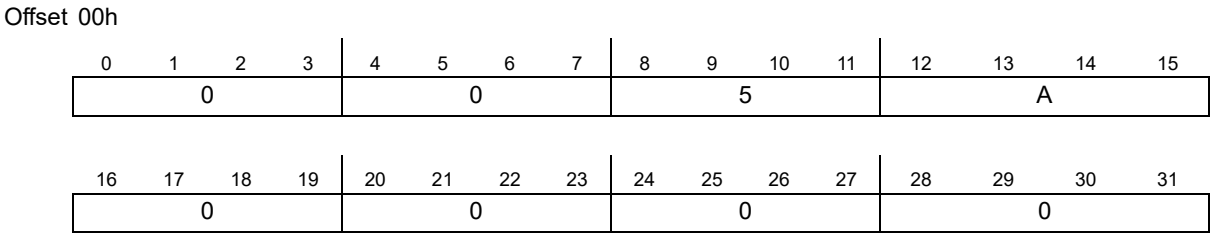


Figure 17. Boot header configuration

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 and make the cores jump to their respective reset vectors. At this point the BAF has finished its operation.

6.9.1.3 LINFlexD_0 and M_CAN_1 pin configuration

Table 54 shows the device pin configurations for LINFlexD_0 and M_CAN_1 modules.

Table 54. M_CAN_1 and LINFlexD_0 pin configuration

Pads	Reset Function	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[10]	GPIO Input buffer off Output buffer off	M_CAN_1 Tx	Output:— Push/Pull medium drive no pullup/pulldown	M_CAN_1 Tx	LINFlexD_0 Tx
PA[11]	GPIO Input buffer off Output buffer off	LINFlexD_0 Rx and M_CAN_1 Rx	Input:— pull-up hysteresis TTL voltage level	M_CAN_1 Rx	LINFlexD_0 Rx

6.9.2 System Status and Configuration Module (SSCM) configuration

This section summarizes the System Status and Configuration Module (SSCM) configuration in the SPC572Lx. For a comprehensive description of the SSCM, refer to the SSCM's dedicated chapter.

6.9.2.1 SSCM instantiation

There is one SSCM instance in the SPC572Lx device. The SSCM is part of the *Chapter 7: Reset and Boot*.

6.9.2.2 Device-specific features

The SSCM will set the reset vector for the Core to execute from the BAF (*Chapter 52: Boot Assist Flash (BAF)*). The BAF allows boot from Flash and external booting from the LINFlexD port without autobaud.

The device-specific reset values for the SSCM_MEMCONFIG register are shown in *Table 55*.

Table 55. SSCM_MEMCONFIG reset values

Field	Field definition	Value
MREV	Minor Revision	Minor silicon version(1)
JPIN	JTAG Part ID	302h(2)

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

6.9.2.3 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 67: Register Protection (REG_PROT)*) for a detailed description of the register protection mechanism.

6.9.3 Safety modules

6.9.3.1 Register protection (REG_PROT) configuration

Some modules on this device include a built-in register protection mechanism that can be used to override the normal write permissions associated with individual registers in the module. This mechanism is software controlled and can be dynamically reconfigured at runtime or made effective until device reset. For a detailed explanation, see the *Chapter 67: Register Protection (REG_PROT)*.

Table 56. Reference links to related information

Topic	Related module	Reference
System memory map	—	*Chapter 5: Memory map*
SIUL2 registers	SIUL2	*Chapter 13: System Integration Unit Lite2 (SIUL2)*

Table 56. Reference links to related information(Continued)

Topic	Related module	Reference
CMU registers	CMU, CMUIOP	*Chapter 23: Clock Monitor Unit (CMU)*
PMC Digital Interface (PMC_DIG) registers	PMC_DIG	*Chapter 54: Power Management Controller digital interface (PMC_dig)*
PLL Digital Interface (PLL_DIG) registers	PLLDIG	*Chapter 22: PLL Digital Interface (PLLDIG)*
Oscillator Digital Interface (OSC_DIG) registers	XOSC	*Chapter 25: OSC Digital Interface (XOSC)*
Mode Entry (MC_ME) registers	MC_ME	*Chapter 56: Mode Entry Module (MC_ME)*
Clock Generation Module (ME_CGM) registers	MC_CGM	*Chapter 24: Clock Generation Module (MC_CGM)*
Reset Generation Module (MC_RGM) registers	MC_RGM	*Chapter 51: Reset Generation Module (MC_RGM)*
System Status and Control Module (SSCM) registers	SSCM	*Chapter 53: System Status and Configuration Module (SSCM)*
JTAG Master (JTAGM) registers	JTAGM	

6.9.3.2 SIUL2 protected registers

Table 57 lists the SIUL2 registers that can be protected.

Table 57. 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
MCSR 0–15	32	0x0240	32 (×16)
MCSR 24–27	32	—(1)	32 (×4)
MSCR 32–47	32	—(1)	32 (×16)
MCSR 49	32	—(1)	32
MCSR 51–54	32	—(1)	32 (×4)
MCSR 56–76	32	—(1)	32 (×21)
MCSR 109	32	—(1)	32
MCSR 512–527	32	—(1)	32 (×8)
MCSR 582-583	32	—(1)	32 (×2)
MCSR 608	32	—(1)	32

Table 57. Protected SIUL2 registers(Continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
MCSR 612	32	—(1)	32
MCSR 624	32	—(1)	32
MCSR 628	32	—(1)	32
MCSR 638-639	32	—(1)	32
MCSR 643	32	—(1)	32
MCSR 656-687	32	—(1)	32 (×22)
MCSR 758	32	—(1)	32
MCSR 759	32	—(1)	32
MCSR 936-967	32	—(1)	32 (×32)
SIUL2_GPDO0–511	8	0x1300	8 (×512)
SIUL2_PGPDO0–31	16	0x1700	16 (×32)
SIUL2_MPGPDO0–31	32	0x1780	32 (×32)

1. See the module memory map in the SIUL2 chapter for address offset.

6.9.3.3 Clock Monitor Unit (CMUIOP) protected registers

Table 58 lists the registers that can be protected for the CMU_IOP.

Table 58. Protected CMUIOP registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CMU_CSR	32	0x0	8 (byte 0)
CMU_HFREFR_A	32	0x0008	32
CMU_LFREFR_A	32	0x000C	32
CMU_MDR	32	0x0018	32

6.9.3.4 Power Management Controller Digital Interface (PMC_dig)

Table 59 lists the registers that can be protected for the power management controller.

Table 59. Protected PMC_dig registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
REE_VD3	32	0x0034	32
RES_VD3	32	0x0038	32
REE_VD4	32	0x0044	32
RES_VD4	32	0x0048	32
REE_VD7	32	0x0074	32

Table 59. Protected PMC_dig registers(Continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
RES_VD7	32	0x0078	32
REE_VD9	32	0x0094	32
RES_VD9	32	0x098	32
REE_VD10	32	0x0A4	32
RES_VD10	32	0x0A8	32
REE_TD	32	0x0304	32
RES_TD	32	0x0308	32
CTL_TD	32	0x030C	32
MREG_CTRL	8	0x240	8
VD_UTST	8	0x340	8
ADC_CH	8	0x344	8

6.9.3.5 PLL Digital Interface (PLLDIG) protected registers

Table 60 lists the PLLDIG registers that can be protected.

Table 60. Protected PLLDIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PLL0CR — PLL0 Control Register	32	0x0000	8
PLL0DIVR — PLL0 Divider Register	32	0x0008	32

6.9.3.6 Oscillator Digital Interface (XOSC) protected registers

Table 61 lists the XOSC registers that can be protected.

Table 61. Protected XOSC registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
OSC_CTL	32	0x0000	32

6.9.3.7 Mode Entry (MC_ME) protected registers

Table 62 lists the MC_ME registers that can be protected.

Table 62. Protected MC_ME registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ME_MCTL	32	0x0004	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_RUN_PC0-7	32	0x080 – 0x09C	32
ME_PCTL 0 – 255	8	0x0C4 – 0x1BF	8
ME_CCTL0-7	16	0x1C4-1D2	16

6.9.3.8 Clock Generation Module (MC_CGM) protected registers

Table 63 lists the MC_CGM registers that can be protected.

Table 63. Protected MC_CGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CGM_SC_DC[0..3]	32	0x07E8-0x7F4	32
CGM_AC0..15_SC	32	0x800-0x9E0	8 (byte 3)
CGM_AC0..15_DC(0..3)	32	0x808..814-0x9E8..9F4	32

6.9.3.9 Reset Generation Module (MC_RGM) protected registers

Table 64 lists the MC_RGM registers that can be protected.

Table 64. 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

Table 64. Protected MC_RGM registers(Continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
RGM_FESS	32	0x340	32
RGM_PRST0–7	32	0x610–62C	32

6.9.3.10 System Status and Control Module (SSCM) protected registers

Table 65 lists the SSCM registers that can be protected.

Table 65. Protected SSCM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ERROR	16	0x0006	16

6.9.3.11 JTAG Master (JTAGM) protected registers

Table 66 lists the JTAGM registers that can be protected.

Table 66. Protected JTAGM registers

Register	Register Size (bits)	Offset from Module Base Address	Protected Size (bits)
JTAGM_MCR	32	0x00	32

6.10 Security modules

6.10.1 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 SPC572Lx MCU. Up to four levels of password protection can be implemented for each flash block.

Note: The registers discussed in this section are initialized via DCF records, so the reset values depend on user settings.

See the Password and Device Security Module (PASS) chapter for further details.

6.10.1.1 PASS_LOCK*n*_*PGn* register bit mapping

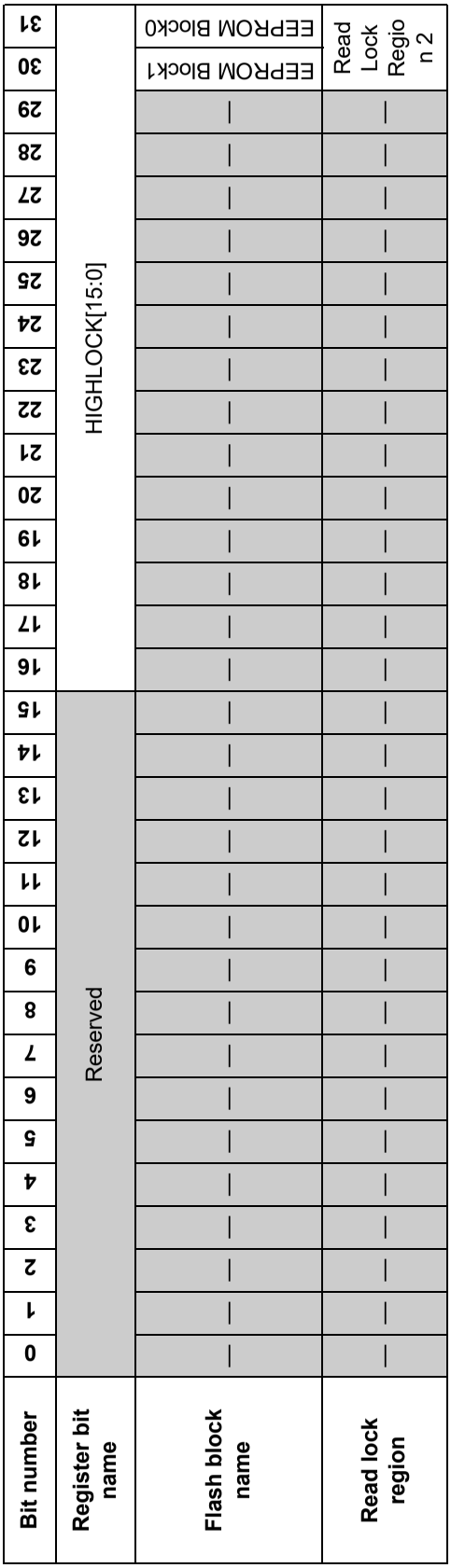
Each password group defined in the DCF records has a set of four LOCKn registers association with it—PASS_LOCK*n*_PG*n*. The bits of these registers are associated with specific flash blocks. The mapping is shown in the following figures.

Note: The PASS module has additional mapping in the LOCK3 register and defines read locking regions. See the PASS chapter for details.

Figure 18. PASS_LOCK0_PG*n* register

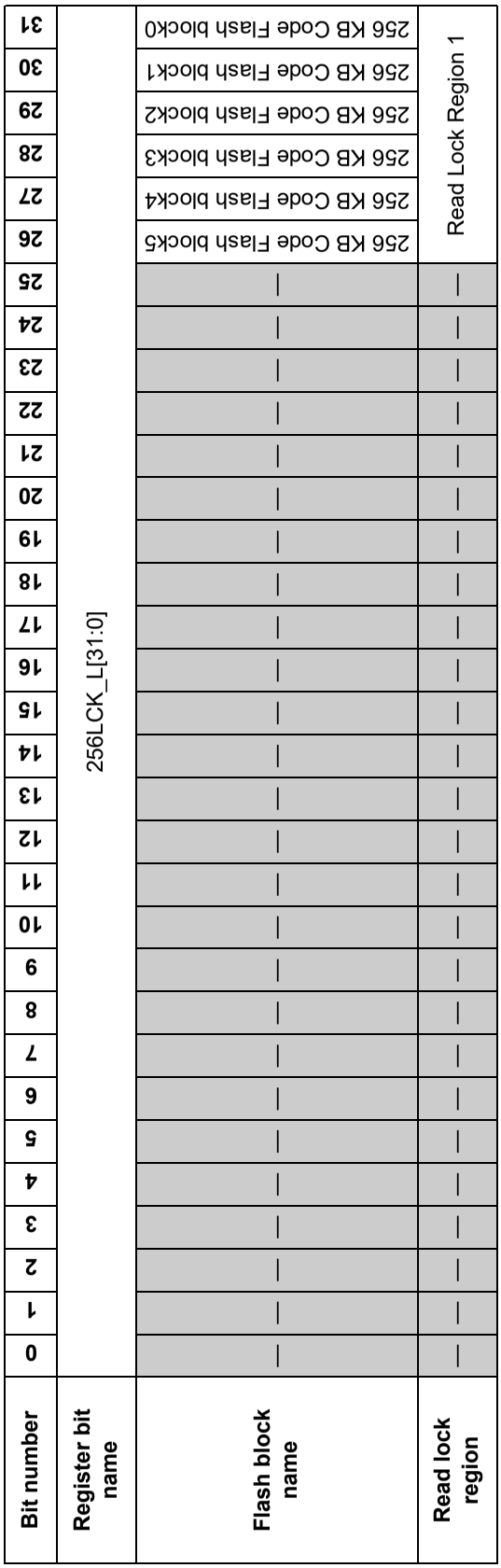
Bit number	Register field	Flash block name	Read lock region
0	TSLOCK	UTEST NVM Block Space 8 KB	Read Lock Region 0
1	Reserved		
2	LOWLOCK[13:0]		
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15		8 KB BAF block0	
16	MIDLOCK[15:0]		
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			

Figure 19. PASS_LOCK1_PG*n* register



Bit number	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
Register bit name	HIGHLOCK[15:0]																Reserved															
Flash block name	EEPROM Block1	EEPROM Block0																														
Read lock region	Read Lock Region 2																															

Figure 20. PASS_LOCK2_PG*n* register



Bit number	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
Register bit name	256LCK_L[31:0]																															
Flash block name	256 KB Code Flash block0	256 KB Code Flash block1	256 KB Code Flash block2	256 KB Code Flash block3	256 KB Code Flash block4	256 KB Code Flash block5																										
Read lock region	Read Lock Region 1																															

Figure 21. PASS_LOCK3_PG*n* register

Bit number	Register bit name	Flash block name
0	PGL	Password Group Lock
1	DBL	Debug Interface Lock
2	MO	Master Only
3	Reserved	–
4	MSTR	Master Access
5		
6		
7		
8	Reserved	–
9		
10		
11		
12		
13	RL2	EEPROM Data Flash Read Lock
14	RL1	Code Flash Read Lock
15	RL0	UTEST Read Lock
16	256KLOCK[15:0]	–
17		–
18		–
19		–
20		–
21		–
22		–
23		–
24		–
25		–
26		–
27		–
28		–
29		–
30		–
31		–

6.10.1.2 PASS Module DCF Records (includes life cycle state and censorship)

The records required to configure the initial values for the PASS module password group registers are shown in *Table 67*. These values provide the following capabilities:

- Set flash memory to censored/uncensored state (censorship also depends on lifecycle)
- Override the basic flash write protection for flash memory blocks
- Set levels of password-based write protection (up to four 256-bit passwords can be required) for individual blocks of flash memory
- Set levels of password-based read protection (up to four 256-bit passwords can be required) for up to five flash memory Read Locking Regions
- Set levels of password-based protection (up to four 256-bit passwords can be required) for the debug part.

Most DCF records represent an initial value or reset value for a module register. See the relevant register documentation for details on values that can be assigned.

Table 67. 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	00101001–00101011	Reserved		
000_0000_0000_1000	0000000	00101100	Censorship(1)	16	1
000_0000_0000_1000	0000000	00101101–00101111	Reserved		
000_0000_0000_1000	0000000	00110001–00111111	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
000_0000_0000_1000	0000000	01001110	LOCK2_PG3(2)	32	1

Table 67. PASS DCF records(Continued)

DCF CS[14:0]	DCF address (binary)		DCF client description	Number of	
	[16:10]	[9:2]		Valid bits	DCF records
000_0000_0000_1000	0000000	01001111	LOCK3_PG3[2]	32	1
000_0000_0000_1000	0000000	01001100–11111111	Reserved		

1. See *Section 68.5.4: Censoring and uncensoring the device* for values
2. See *Section 6.10.1.1: PASS_LOCKn_PGn register bit mapping* for values

6.11 Debug and calibration architecture

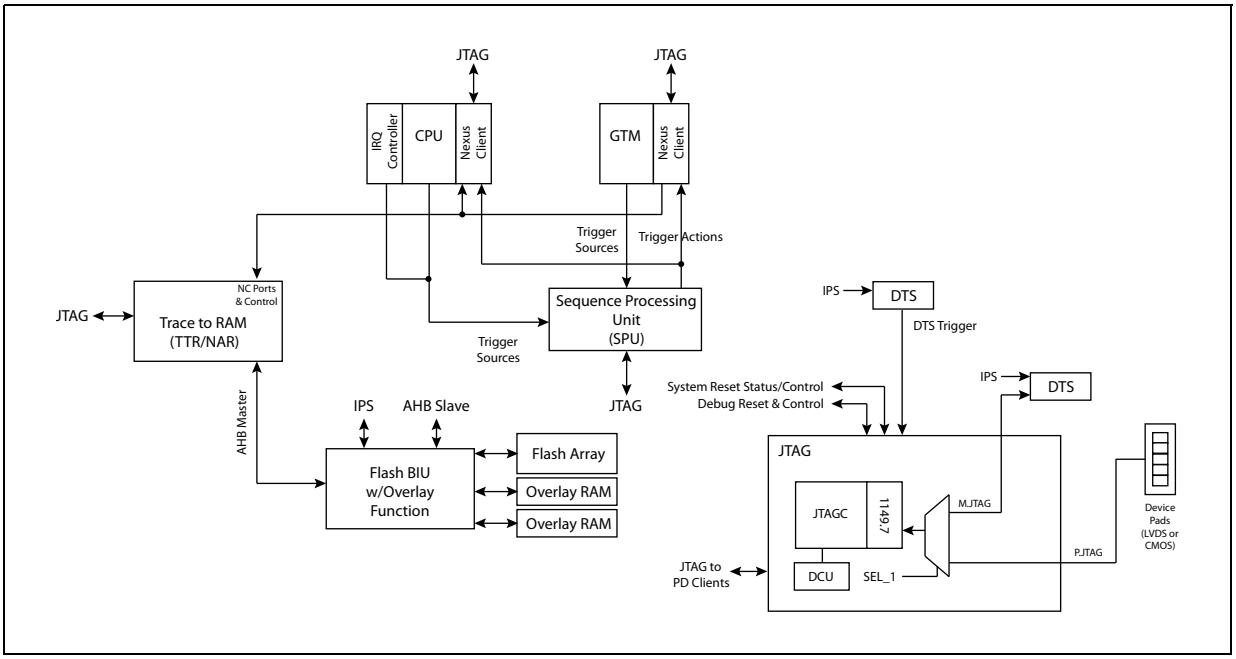
SPC572Lx microcontrollers have a comprehensive feature set to assist debug and calibration.

Figure 22 shows the top-level block diagram of the SPC572Lx production die. The figure shows the general interconnection of functional debug modules.

Acronyms used in the block diagram include:

- DCI—Debug and Calibration Interface
- JTAGC—JTAG Controller
- M.JTAG—Mastered JTAG
- P.JTAG—Peripheral JTAG
- I.JTAG—Inter-die JTAG
- DCU—Debug Control Unit
- DTS—Development Tool Semaphore
- JTAGM—JTAG Master
- SPU—Sequence Processing Unit
- NAR—Nexus Aurora Router
- Nexus RWA—Nexus Read/Write Access client

Figure 22. Debug and calibration architecture for SPC572Lx



7 Reset and Boot

7.1 Introduction

Resetting the SPC572Lx involves an entire process starting with the application of power to fetching the first instruction of the user's application code.

The processor core on the SPC572Lx is based on the e200z215An3 architecture.

Memory elements involved in the boot-up process include:

- UTEST flash memory.
- TEST flash memory.
- Boot Assist Flash (BAF).

7.1.1 Boot Assist Flash

The Boot Assist Flash (BAF) contains factory code to facilitate the device boot-up procedure.

If the BAF code does not locate a valid boot header, it runs the Serial Boot Loader which receives files from an external serial link controlled by the internal LINFlexD module. The received file is placed in RAM and program execution begins at the start address specified in the received file.

The Serial Boot Loader is only an option in the early stages of the device life cycle. In later stages (for example *FIELD*) the device waits for the watchdog to reset the device if no boot header is found.

7.1.2 TEST flash memory block

The TEST flash memory block contains Device Configuration Format (DCF) records used to hold trim values and other variables, as well as general device configuration information.

The trim values are for:

- Adjusting low-voltage and high-voltage detect circuit trip points.
- Temperature sensor adjustments.
- Power-on reset voltage trip point.
- Analog-to-digital adjustments.
- Internal RC oscillator (IRCOSC) trim values.

Note: The DCF records are written by the device manufacturer and programmed into TEST flash memory during production testing. Further programming of the TEST flash is disabled at the end of the factory test cycle.

7.1.3 UTEST flash memory block

The UTEST flash memory block also contains DCF records. Some UTEST DCF records are written by the factory and programmed during production testing. Others are written by the end user and programmed at the same time application code is programmed into the flash memory. UTEST DCF records are used to set up various control and configuration registers, including default memory configuration.

7.1.4 Boot header

A boot header is used to supply the start address for code execution for the Core. It is written by the application programmer and loaded into specific locations in flash memory. The boot header contains the reset vectors for the core.

7.2 Modules used in reset sequence

The modules involved in the SPC572Lx reset sequence are:

- Power Management Controller (PMC).
- Reset Generation Module (MC_RGM).
- Mode Entry Module (MC_ME).
- System Status and Configuration Module (SSCM).

7.2.1 Power Management Controller

The Power Management Controller (PMC) controls and monitors:

- Its own supply voltage.
- The supply voltages to all the high- and low-voltage detect circuits.
- The trip points for all the high- and low-voltage detect circuits.
- The power supplies and reference voltages to the Analog-to-Digital Converter.
- The major power supplies to the SPC572Lx.

7.2.2 Reset Generation Module

The Reset Generation Module (MC_RGM) is a complex state machine that begins sequencing the SPC572Lx through the initial steps of the reset process. The MC_RGM does not execute program code, it is a state machine that centralizes the different reset sources and manages the reset sequence. Reset sources are organized into two categories: destructive and functional.

See *Figure 23* for more information on the MC_RGM reset sequence.

7.2.2.1 Destructive resets

A destructive reset source is related to a critical error or dysfunction, usually caused by a hardware malfunction. When a destructive reset event occurs, software recovery is not possible and the contents of memory are assumed unknown. A full device reset sequence starting from PHASE0 is therefore applied to the device, ensuring a safe startup state for both digital and analog modules.

Examples of destructive resets are:

- Power-on reset.
- Low voltage detection.

7.2.2.2 Functional resets

A functional reset source is related to a less critical error or dysfunction usually not associated with hardware malfunction. When a functional reset event occurs, a partial reset sequence is applied to the device starting with PHASE1: most digital modules are reset

normally and the state of analog modules, specific digital modules (e.g., debug and flash memory modules) and system memory content is preserved.

Examples of functional resets are:

- External reset.
- Machine check.
- Software reset from mode entry.
- Boundary scan instructions.

7.2.3 Mode Entry module

The Mode Entry module (MC_ME) is responsible for delivering initial values to many registers, including the reset vectors to the SPC572Lx core.

During the power-up sequence, the System Status and Configuration Module (SSCM) searches for a boot header at defined locations in the flash memory, derives the necessary reset vectors from the boot header and then writes these vectors to their respective locations in the Mode Entry module.

The SSCM also interprets the DCF records which contain further information that is written to the MC_ME. As the boot-up sequence progresses, the SSCM instructs the MC_ME to transfer the reset vectors to the core.

Even though the reset vectors are in the boot header located in flash memory, the Mode Entry module always provides them to the CPU cores.

7.2.4 System Status and Configuration Module

During the reset sequence, the Reset Generation Module (MC_RGM) enables the SSCM state machine which reads the DCF records in the TEST and UTEST flash memory areas for device setup information to be written to various registers in the SPC572Lx.

The DCF records are primarily for setting up the memory and providing initial device configuration values.

The SSCM continues with the reset or boot-up sequence by locating boot vectors for the core. Once the SSCM locates the boot header, the SSCM writes the start address information for the CPU core to special locations in the Mode Entry Module (MC_ME). The MC_ME feeds a reset vector to the core which begins program execution at the specified address.

The boot-up sequence ends when the SSCM starts up the CPU.

7.3 Reset sequence

The power-up reset sequence always begins with the application of power and follows different sequences depending on the condition of the SPC572Lx device and whether various modes are enabled from settings in the DCF records. For instance, the SPC572Lx device enters Serial Boot mode (the Serial Boot Loader receives startup code and begins program execution) if a valid boot header file is not found.

Another reset outcome is that the CPU is properly started and begins executing program code.

7.3.1 Power-on and the Reset Generation Module

When power is applied to the SPC572Lx, the Reset Generation Module (MC_RGM) advances the device through a series of steps shown in *Figure 23*.

The Reset Generation Module (MC_RGM) starts the System Status and Control Module (SSCM), which continues the boot-up process detailed in *Figure 24* through to *Figure 27* after the MC_RGM enters the IDLE state.

Figure 23. Reset Generation Module Reset Sequence

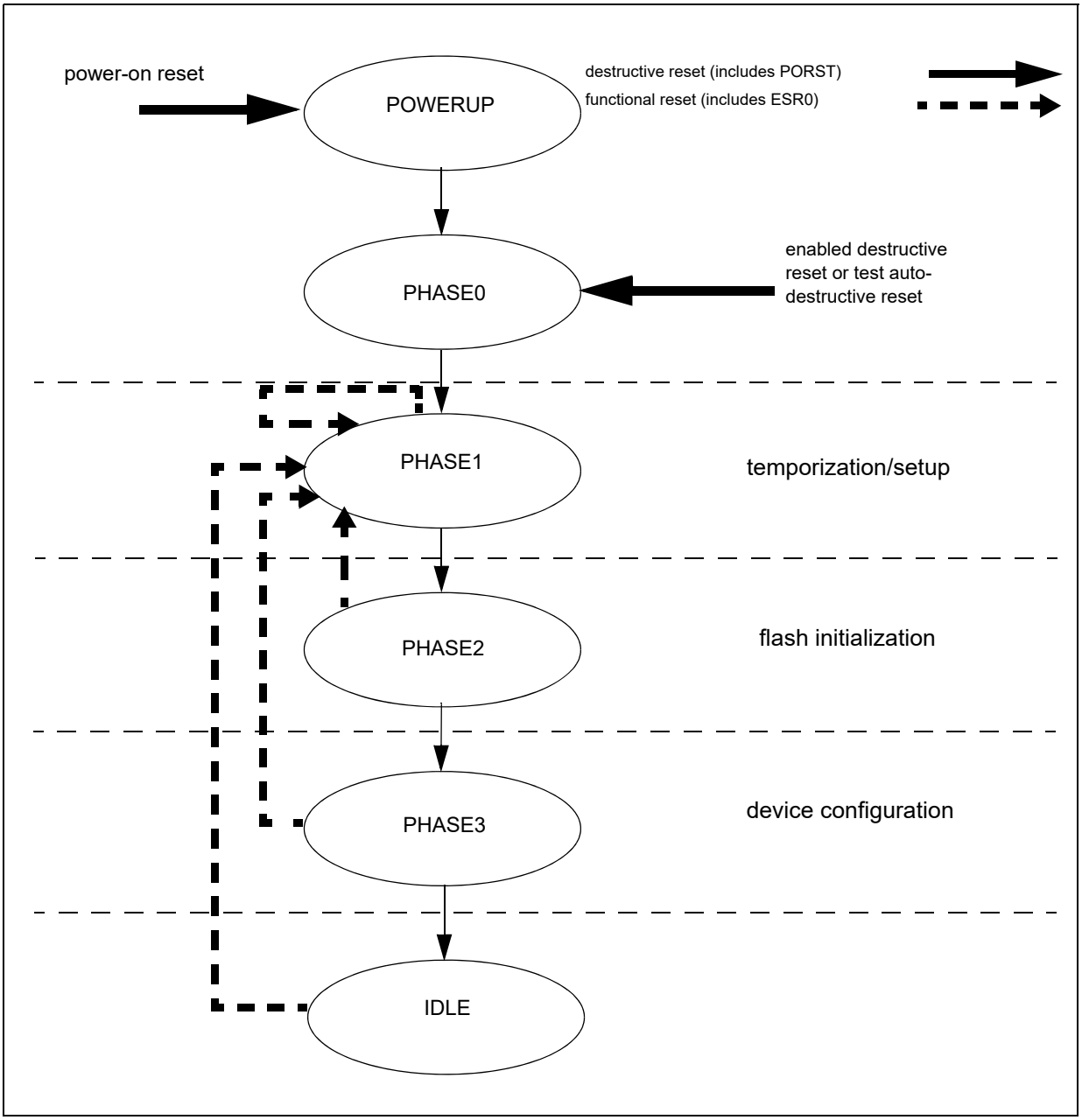


Table 68. Module status during reset phases

Module	PHASE				
	PHASE0	PHASE1	PHASE2	PHASE3	IDLE
IRCOSC	ON	ON	ON	ON	ON
PMC	ON	ON	ON	ON	ON
MC_RGM	ON(1)	ON	ON	ON	ON
System watchdog	RESET	RESET	ON	RESET	ON(2)
Flash module	RESET	RESET	ON	ON	ON
XOSC	RESET	ON(2)	ON(2)	ON(2)	ON(2)
DEVICE CONFIG	RESET	HOLD	HOLD	HOLD	HOLD
SSCM	RESET	RESET	RESET	ON	ON
Core	RESET	RESET	RESET	RESET	ON
PIT_1	RESET	IDLE/ON	IDLE/ON	IDLE/ON	IDLE/ON
Other modules	RESET	RESET	RESET	RESET	ON
				Short functional reset	
		functional reset			
	destructive reset				

1. RESET only during POR
2. Depending on flash configuration

7.3.2 Power-up phase: power stabilization

When a power-on reset event occurs (initial application of power), the reset generation module (MC_RGM) causes the device to enter the power-up phase of the reset sequence (See *Figure 23*). The only method to enter the power-up phase is from a power-on reset event.

The device remains in the power-up phase until the Power Management Controller module (PMC) determines that all power supplies are in their respective operational ranges.

The supplies for initial configuration are:

- VDD_HV_CORE: high voltage supply for core module, including power management unit, internal RC oscillator (IRCOSC), external oscillator (XOSC) driver.
- VDD_LV_CORE: low voltage supply for digital module.
- VDD_HV_FLA: high voltage supply for flash module.
- VDD_LV_FLA: low voltage supply for flash module.
- VDD_HV_IO_IM: high voltage supply associated with system pins (PORST, ESR0, TEST Mode).

Peripheral supplies such as VDD_HV_IO_JTAG or VDD_HV_ADC do not gate the power-up process. The low- and high-voltage detect circuits for these and other power supplies are

managed by the PMC, which provides enable and status bits for these voltage detection circuits.

The PMC sends a signal to the Reset Generation Module when power stabilization has been achieved. The Reset Generation Module then advances to the PHASE0 state (see *Figure 23*). During the power-up phase, the PMC does the following:

- Drives to a logic ‘0’ all LVDxx (low-voltage detect) and HVDxx (high-voltage detect) signals monitored by the Power Management Control (PMC) module.
- Monitors its own power supply voltage (VDD_HV_PMC) to determine that the applied voltage is within a specified range.
- Monitors the core power supply voltage (VDD_LV_CORE) to determine that the applied voltage is within a specified range.
- Monitors the low- and high-voltage detect circuit power supply voltage to determine that the applied voltage is within a specified range.
- Holds the outputs of the internal low- and high-voltage detect circuits at a ground state until the PMC determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Enables all low- and high-voltage detect circuits once all power supplies are at their functional levels.
- Begins to monitor the various power supplies using the high- and low-voltage detect circuits.
- Provides status information to the Reset Generation Module (MC_RGM) indicating whether or not power is properly applied.

To exit power-up and enter PHASE0:

- All enabled ‘destructive’ resets must be processed.
- All power supplies must be at their functional voltages.
- The PMC must signal the MC_RGM that all power supplies are at the required levels.

7.3.3 *PHASE0* Phase: analog supply initial configuration

This phase is entered:

- On exit from power-up phase.
- On PORST pin falling edge detection except for the initial power-up sequence.
- Immediately from PHASE1, PHASE2, PHASE3 or IDLE, on the occurrence of a PORST falling edge event other than power-on reset.

During PHASE0:

- All digital modules are reset, including safety, security, and test modules.
- All trimming bits for analog modules are reset.
- Startup of the internal analog modules (PMC, IRCOSC, and I/Os) begins:
  - PMC determines that proper voltages are applied.
  - I/O pins are ensured correctly configured: outputs are driven to known levels, inputs ignored.

To exit PHASE0 and enter PHASE1:

- All enabled destructive resets must be processed.
- Gating PHASE0 must be released. See *Table 74: POR and voltage monitors description*.
- All processes initiated in PHASE0 must be completed. This generally includes startup of the internal analog modules: PMC, IRCOSC, and I/Os.

This phase is typically completed in less than 20 µs. The device is guaranteed to remain in this phase for at least 0.5 µs.

Note: The core voltage must rise above LVD108 upper threshold to exit PHASE0. This ensures significant hysteresis on low voltage supply avoiding spurious low voltage detection during the power-up process.

7.3.4 *PHASE1*: temporization and monitoring setup

This phase is entered on exit from PHASE0.

During PHASE1:

- Initial configuration of the watchdog to monitor the reset sequence.
- Temporization for reset signal propagation in case of reset.

This phase is completed in less than 1 µs and corresponds to eight IRCOSC clock cycles.

Test modules are available during PHASE1, but their functionality is limited by the security module (no security waiver granted at this phase).

All PHASE1 tasks must be completed before entering PHASE2.

7.3.5 *PHASE2*: flash initial configuration

This phase is entered on exit from PHASE1.

During PHASE2:

- Reset is released to the flash memory module by the Reset Generation Module.
- The flash memory module starts its initial configuration process.
- The flash memory initialization is performed by a state machine internal to the flash module.
- The system watchdog starts monitoring flash memory configuration execution so that a reset occurs in the event of configuration failure.

Note: System Watchdog is configured to generate destructive reset in case of flash memory failure during boot process.

All PHASE2 tasks must be completed before entering PHASE3.

This phase is completed in less than 200 µs.

7.3.6 *PHASE3*: device configuration

This phase is entered on exit from PHASE2.

During PHASE3, the System Status and Configuration Module (SSCM) starts by retrieving the Device Configuration Format (DCF) record from the TEST and UTEST flash memory areas, and uses the configuration and initialization information therein to:

- Configure the initial memory map of the device.
- Transfer reset vectors for the core to the Mode Entry Module (MC_ME).
- Write trim values for analog modules including the Analog-to-Digital Converter, low-voltage detect circuits, and the temperature sensor to their respective registers.

The DCF record programmed into the TEST flash memory area is generally written by the manufacturer. The portion of the DCF record programmed into the UTEST flash memory is usually written by the customer, although there is a default DCF record programmed into both the TEST and the UTEST flash memory areas during device production testing. In general, you write your own DCF record for the UTEST area to replace the default DCF.

The DCF record also provides configuration information for the following:

- Trimming of the analog modules (for example, voltage regulator, voltage detectors, I/Os, and oscillator)
  - Trim values are in DCF records in the TEST flash memory area.
  - Trim values are determined during production test and programmed into the TEST flash memory area.
- Device life cycle information:
  - Censorship configuration.
  - Debug security configuration.
  - Functional security configuration.
- Application configuration bits:
  - Watchdog configuration.
  - Safety execution directives.
  - Boot Assist Flash options.
  - Oscillator startup during reset sequence configuration.

To exit PHASE3: All processes that need to be completed in PHASE3 must be finished.

This phase depends on actual trim values, especially those used for safety execution directives. This device configuration is maintained until the next internal power-on reset. Each time a destructive reset event occurs, the SSCM takes values from the DCF record and writes them to the applicable registers.

Note: It is possible to maintain the device in PHASE3 by forcing either the PORST or ESR0 pin low. This can be used for debugging purposes by allowing external test/development equipment to connect through JTAG to the internal debugger.

7.3.7 *IDLE* Phase

This is the Reset Generation Module's final phase following completion of PHASE3.

Once the IDLE phase is signaled, the System Status and Control Module (started in PHASE3) continues with the system boot-up sequence while the MC_RGM waits for new events that trigger a reset sequence.

7.4 MC_RGM passes boot sequence control to the System Status and Control Module

A flow chart of the boot-up sequence is shown in *Figure 24*, *Figure 25*, *Figure 26*, and *Figure 27*.

7.4.1 Reset sequence flow based on initial device condition

The boot-up sequence takes different paths depending on which modules are enabled and which header files are present. For instance, a device shipped from the factory has pre-installed programs in the Boot Assist Flash and the Boot Assist ROM. DCF records are programmed into the TEST flash memory and the UTEST flash memory.

- Boot Assist Flash factory code cannot be modified by user.
- Boot Assist ROM factory code cannot be modified by user.
- TEST DCF record cannot be modified by user.
- UTEST flash memory contains a default DCF record which can be replaced with a user file.

Programmed SPC572Lx flash memory contains the CPU boot header file and the user application. The boot header is used by the boot-up sequence flow managed by the SSCM module.

7.4.2 Possible boot-up sequences

7.4.2.1 Enter JTAG test mode

- Used for JTAG testing (non-user mode)
- Interface to JTAG port Development Equipment for Code Development

7.4.2.2 Enable the Boot CPU

- Normal operation for the boot CPU.

7.4.2.3 Enable Boot CPU and enter serial boot mode if no application code is present

- Mode used to download program code and place it in RAM.
- Mode used to start downloaded application program.

The following sections detail the exact flow through the reset sequence to achieve the various outcomes.

7.4.3 Sequence from MC_RGM IDLE to Boot CPU running code

The CPU starts executing application code.

7.4.3.1 Assumptions

- Boot Assist Flash bypass mode is not enabled (execute BAF code).
- Application code for the Boot CPU is programmed into flash memory.
- A valid boot header has been programmed into flash memory.
  - Boot Header_ID
  - Boot CPU selection
  - Boot CPU start address (address of first instruction in application code)
  - CPU reset vector
- The MC_RGM has passed control to the SSCM during the IDLE state.

7.4.3.2 Reset Generation Module has entered idle state (state 1 and 2)

Once the IDLE phase is signaled, the System Status and Control Module (started in PHASE3) continues with the system boot-up sequence while the MC_RGM waits for new events that trigger a reset sequence.

At this point:

- The data in the DCF record has been written to the appropriate registers.
- All the trim values for the analog portions of the chip have been installed in their proper locations.

7.4.3.3 Is a JTAG request pending? (state A)

The SSCM is now in charge of the boot-up sequence. It begins by determining whether a JTAG request is pending. The JTAG test circuitry is for production test and used by software development systems. The JTAG mode has no user functions and should never be used in a user application. System designers must ensure that the external JTAG request pins cannot inadvertently cause a JTAG test request.

7.4.3.4 Is the Boot Assist Flash bypass mode enabled? (state B)

The SSCM examines a status bit to determine whether the Boot Assist Flash bypass mode is enabled. If this mode is enabled, any eventual code in the BAF is not executed. The BAF bypass mode is enabled/disabled using a DCF record.

The DCF record supplied by the factory disables the bypass mode. This means that the code in the Boot Assist Flash is always executed during a boot-up sequence.

The factory BAF code also runs the serial boot loader, setting up pins on the device to act as a UART: receiving a file using a defined format and transferring the program portion of the file to RAM. Once the program is installed, the SSCM begins core execution at the start address specified in the file header.

7.4.3.5 Provide reset vectors to Core (state 21)

During PHASE3 of the Reset Generation Module's sequence, the start address for execution of the BAF code is transferred to the MC_ME.

The SSCM releases reset to the core and turns on the core clock. The BAF code start address is transferred to the core at this time under the control of the SSCM. Execution of the Boot Assist Flash code begins.

7.4.3.6 BAF code execution begins (state 22)

After the reset signal is released to the core, BAF code execution begins.

7.4.3.7 The Core executes code to apply device setting (state 23)

The core executes BAF code written by the manufacturer to apply specific device settings to the SPC572Lx.

The BAF code begins by initializing registers and applying device settings. The BAF code, written by the device manufacturer, changes from device to device.

7.4.3.8 Core searches for boot header (state 31)

The core continues code execution by searching for the boot header. In the present example, it is assumed that a valid Boot Header is present.

This is the header file that specifies which Boot CPU starts executing the user application code.

The Boot CPU header is programmed into flash memory at the same time the user application program is programmed into flash memory. The core searches for the boot header, which can be located at one of four addresses:

- 0x100_0000
- 0x104_0000
- 0x108_0000
- 0x10C_0000

A valid Boot Header_ID begins with the value 0x005A.

The boot header structure consists of the following words:

1. Boot Header_ID/Boot_CPU
2. Start address

To indicate a valid boot header, the 16-bit Boot_Header_ID field is written with the value 0x005A.

Is a valid boot header present? (decision point N)

In the present example, a valid boot header is assumed to be present in flash memory.

7.4.3.9 Decode boot CPU (state 32)

The core decodes the boot header and then writes the reset vector to the Mode Entry Module (MC_ME).

7.4.3.10 Lock BAF access (state 33)

Further access to the Boot Assist Flash is locked.

7.4.3.11 Boot from internal flash (state 34)

The MC_ME transfers the reset vector to the CPU and code execution begins at the reset vector.

7.4.4 Path from MC_RGM IDLE to serial boot mode

Here no application program or valid boot header is programmed into flash memory. It is an unusual scenario, but it is shown here for illustrative purposes.

7.4.4.1 Initial conditions

- No valid boot header has been programmed into flash memory.
- BAF bypass mode is NOT enabled (execute BAF code).
- Application code for the core is programmed into the BAF.
- Application code for the Boot CPU is not programmed into flash memory.
- The Life Cycle variable is set to FACTORY, indicating an unprogrammed device.
- The MC_RGM has passed control to the SSCM during the IDLE state.

7.4.4.2 Reset Generation Module has entered idle state (state 1 and 2)

Once the IDLE phase is signaled, the System Status and Control Module (started in PHASE3) continues with the system boot-up sequence while the MC_RGM waits for new events that trigger a reset sequence.

At this point:

- The data in the DCF record has been written to the appropriate registers.
- All the trim values for the analog portions of the chip have been installed in their proper locations.

7.4.4.3 Is a JTAG request pending? (state A)

The SSCM now controls the boot-up sequence and begins by determining if a JTAG request is pending. The JTAG test circuitry is for production test and use by software development systems. The JTAG mode has no user functions and should never be used in a user application. System designers must ensure that the external JTAG request pins cannot inadvertently cause a JTAG test request.

7.4.4.4 Is the Boot Assist Flash bypass mode enabled? (state B)

The SSCM examines a status bit to determine whether the BAF bypass mode is enabled. If enabled, any code in the BAF is not executed. The BAF bypass mode is enabled/disabled using a DCF record.

The DCF record supplied by the factory disables the bypass mode. This means that the code in the BAF is always executed during a boot-up sequence.

The BAF code written by the device manufacturer also runs the serial boot loader. This involves setting up pins on the device to act as a UART: receiving a file using a defined format and transferring the program portion of the file to RAM. Once the program is installed, the SSCM begins the core execution at the start address specified in the header portion of the program file.

7.4.4.5 Provide reset vector to Core (state 21)

During PHASE3 of the Reset Generation Module's sequence, the start address for execution of the BAF code in the Boot Assist Flash is transferred from the DCF record to the MC_ME by the SSCM.

The SSCM releases reset to the core and then turns on the clock to the core. The start address is transferred to the core at this time under the control of the SSCM and the core begins execution of the BAF code.

7.4.4.6 Core begins BAF code execution (state 22)

After the reset signal is released and the respective clocks are turned on to the core, the core begins execution of code in the Boot Assist Flash.

7.4.4.7 Core continues BAF code execution (state 23)

The BAF code begins by initializing registers and applying device settings. The BAF code is written by the device manufacturer and changes from device to device.

7.4.4.8 Core searches for boot header (state 31)

The core continues code execution by searching for the boot header. In the present example, it is assumed that no boot header file is present.

This is the header file for the CPU that starts executing the user application code.

The Boot CPU header is programmed into flash memory at the same time the user application program is programmed into flash memory. The core searches for the boot header, which can be located at one of four addresses:

- 0x100_0000
- 0x104_0000
- 0x108_0000
- 0x10C_0000

A valid Boot Header_ID begins with the value 0x005A.

The boot header structure consists of the following words:

1. Boot Header_ID/Boot_CPU
2. Start address

To indicate a valid boot header, the 16-bit Boot_Header_ID field is written with the value 0x005A.

7.4.4.9 Valid boot header is not found (decision point N)

The core, executing code in the Boot Assist Flash, does not find a valid boot header file in flash memory. Because there is no valid boot header, serial boot mode must be entered to obtain an application program.

Note: This mode receives a program to be executed over a serial port implemented using the LINFlexD port. Once the program is received, using a defined format, the core jumps to the start address of the program and begins execution.

7.4.4.10 Initialize RAM (state 35)

The core, executing code in the Boot Assist Flash, initializes RAM in preparation for entering serial boot mode. Serial boot mode configures the LINFlexD port to receive code from an external source. The LINFlexD port comes in a number of hardware configurations. The SPC572Lx uses LINFlexD, which implements a basic UART. Code in the BAF operates the LINFlexD module and implements the protocol.

7.4.4.11 Enter serial boot mode (state 14)

Figure 27: Boot-up sequence, part D shows the boot loader flow.

7.4.4.12 Life Cycle check indicates FACTORY (decision point G)

There are two possible settings for the Life Cycle parameter:

- FIELD indicates that application code has been programmed into the flash memory of the device. Serial boot mode is exited and IOP enters a 'wait' mode until a watchdog reset occurs.
- FACTORY means that there is no application code programmed into the flash memory. For the present example a setting of FACTORY is assumed.

7.4.4.13 Receive and install application code

Once a serial link is established via the LINFlexD port using a data transmission protocol defined and implemented in the BAF code, the Serial Boot Loader waits to receive the code.

Note: A software watchdog timer is started by the BAF code. A reset occurs if a program is not received before the watchdog timer timeout.

1. Configure LINFlexD module and external pins (state 37).
2. Wait for START WORD (state 38).
3. Receive START WORD (state 39).
4. Receive START ADDRESS (state 40).
5. Receive DOWNLOAD SIZE (state 41).
6. Receive CODE (state 42).
7. Branch to START ADDRESS (state 43).

Application code begins at the received START ADDRESS.

7.4.5 Path from MC_RGM IDLE with Core enabled to watchdog timer timeout

Here the operation of the core is desired but there is no valid boot header file and the Life Cycle variable is set to FIELD, indicating that the device was programmed with application code and the appropriate boot header files.

This path is only taken if a catastrophic event has occurred or the device is improperly programmed.

7.4.5.1 Initial conditions

- Boot Assist Flash bypass mode is not enabled (execute BAF code).
- A valid boot header for the CPU has not been programmed into flash memory.
  - Boot Header_ID
  - Boot CPU selection
  - Boot CPU start address (address of first instruction in application code)
  - CPU reset vector
- The Life Cycle variable is set to FIELD, indicating a programmed device.
- The MC_RGM has passed control to the SSCM during the IDLE state.

7.4.5.2 Reset Generation Module has entered idle state (state 1 and 2)

Once the IDLE phase is signaled, the System Status and Control Module (started in PHASE3) continues with the system boot-up sequence while the MC_RGM waits for new events that trigger a reset sequence.

At this point:

- The data in the DCF record has been written to the appropriate registers.
- All the trim values for the analog portions of the chip have been installed in their proper locations.

7.4.5.3 Is a JTAG request pending? (state A)

The SSCM now controls the boot-up sequence and begins by determining if a JTAG request is pending. The JTAG test circuitry is for production test and use by software development systems. The JTAG mode has no user functions and should never be used in a user application. System designers must ensure that the external JTAG request pins cannot inadvertently cause a JTAG test request.

7.4.5.4 Is the Boot Assist Flash bypass mode enabled? (state B)

The SSCM examines a status bit to determine whether or not the boot assist flash bypass mode is enabled. If this mode is enabled, any eventual code in the Boot Assist Flash is not executed. The BAF bypass mode is enabled/disabled using a DCF record.

The DCF record supplied by the factory disables the bypass mode. This means that the code in the BAF is always executed during a boot-up sequence.

The factory BAF code also runs the serial boot loader, setting up pins on the device to act as a UART: receiving a file using a defined format and transferring the program portion of the file to RAM. Once the program is installed, the SSCM begins core execution at the start address specified in the file header.

7.4.5.5 Provide reset vectors to Core (state 21)

During PHASE3 of the Reset Generation Module's sequence, the start address for execution of the BAF code is transferred to the MC_ME.

The SSCM releases reset to the core and turns on the core clock. The BAF code start address is transferred to the core at this time under the control of the SSCM. Execution of the Boot Assist Flash code begins.

7.4.5.6 Core begins BAF code execution (state 22)

After the reset signal is released and the core clock is turned on, the core begins execution of BAF code.

7.4.5.7 Core continues BAF code execution (state 23)

The BAF code begins by initializing registers and applying device settings. The BAF code is written by the device manufacturer and changes from device to device.

7.4.5.8 Core searches for boot header (state 31)

The core continues code execution by searching for the boot header. In the present example, it is assumed that no boot header file is present.

This is the header file for the CPU that starts executing the user application code.

The Boot CPU header is programmed into flash memory at the same time the user application program is programmed into flash memory. The core searches for the boot header, which can be located at one of four addresses:

- 0x100_0000
- 0x104_0000
- 0x108_0000
- 0x10C_0000

A valid Boot Header_ID begins with the value 0x005A.

The boot header structure consists of the following words:

1. Boot Header_ID/Boot_CPU
2. Start address

To indicate a valid boot header, the 16-bit Boot_Header_ID field is written with the value 0x005A.

7.4.5.9 Valid boot header is not found (decision point N)

The Core, executing code in the BAF, enters serial boot mode to obtain an application program as it does not find a valid boot header file in flash memory.

Note: This mode uses a defined protocol to receive a program over a serial port via the LINFlexD port. The core then jumps to the start address of the program and begins execution.

7.4.5.10 Initialize RAM (state 35)

The core, executing code in the BAF, initializes RAM areas in preparation for serial boot mode. This mode configures the LINFlexD port to receive code from an external source using a basic UART protocol.

7.4.5.11 Enter serial boot mode (state 14)

Figure 27: Boot-up sequence, part D shows the boot loader flow.

7.4.5.12 Life Cycle check indicates FIELD (decision point G)

There are two possible settings for the Life Cycle parameter:

- FIELD indicates that application code has been programmed into the flash memory of the device. Serial boot mode is exited and IOP enters a ‘wait’ mode until a watchdog reset occurs.
- FACTORY means that there is no application code programmed into the flash memory. For the present example a setting of FACTORY is assumed.

7.4.5.13 Stop—wait for watchdog (state 36)

This state is entered when the SPC572Lx’s program flash memory is incorrectly programmed.

Figure 24. Boot-up sequence part A

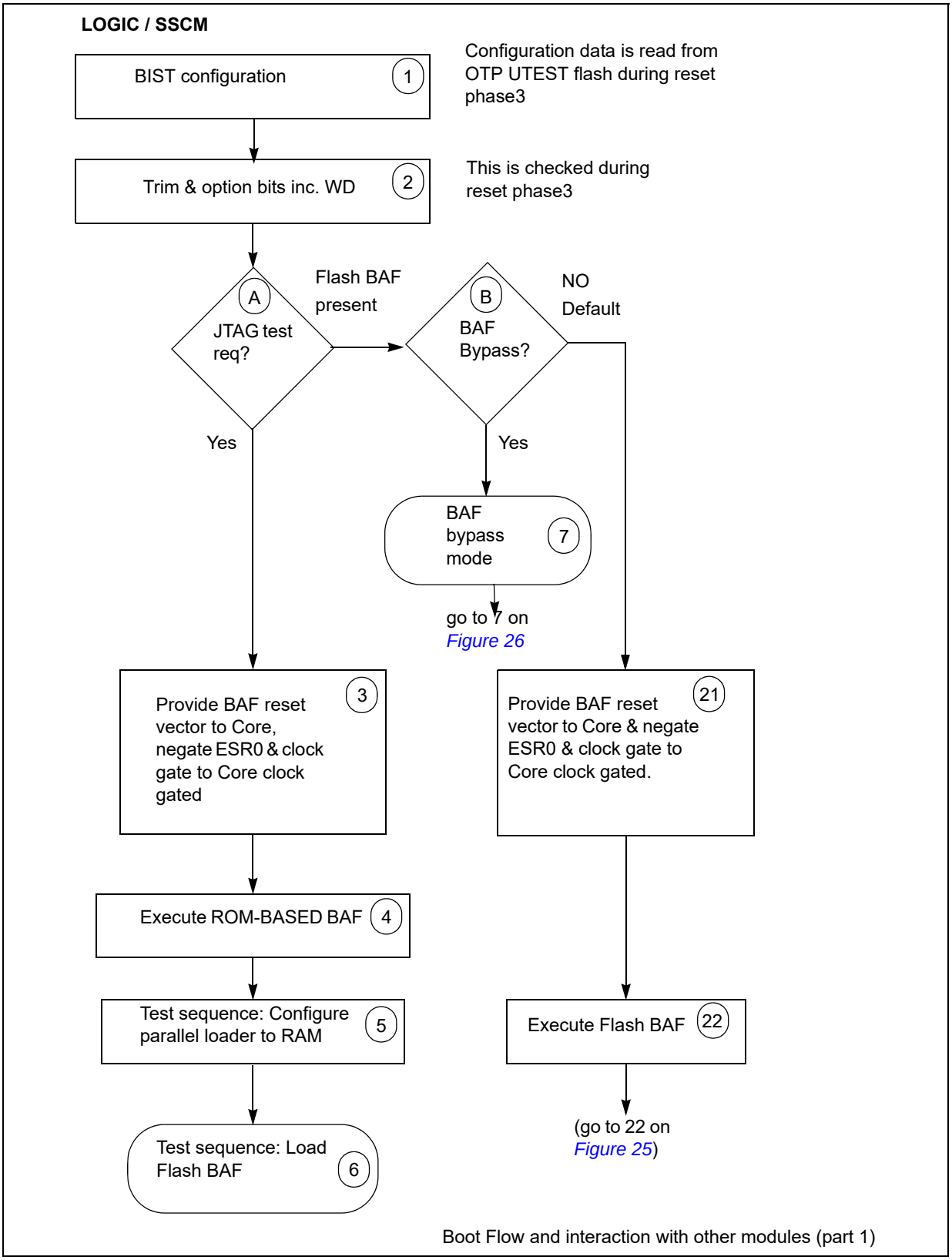


Figure 25. Boot-up sequence part B

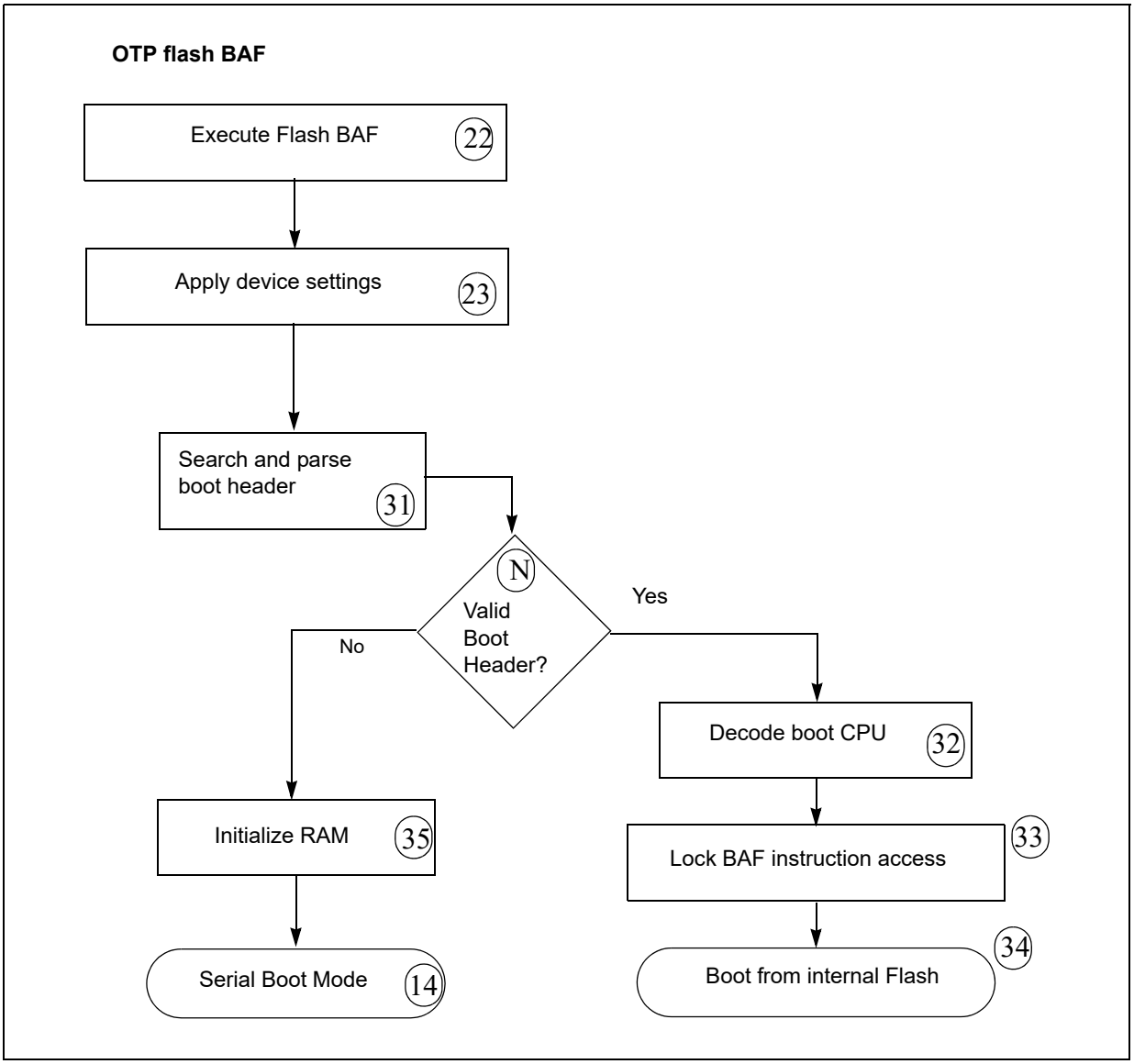


Figure 26. Boot-up sequence, part C

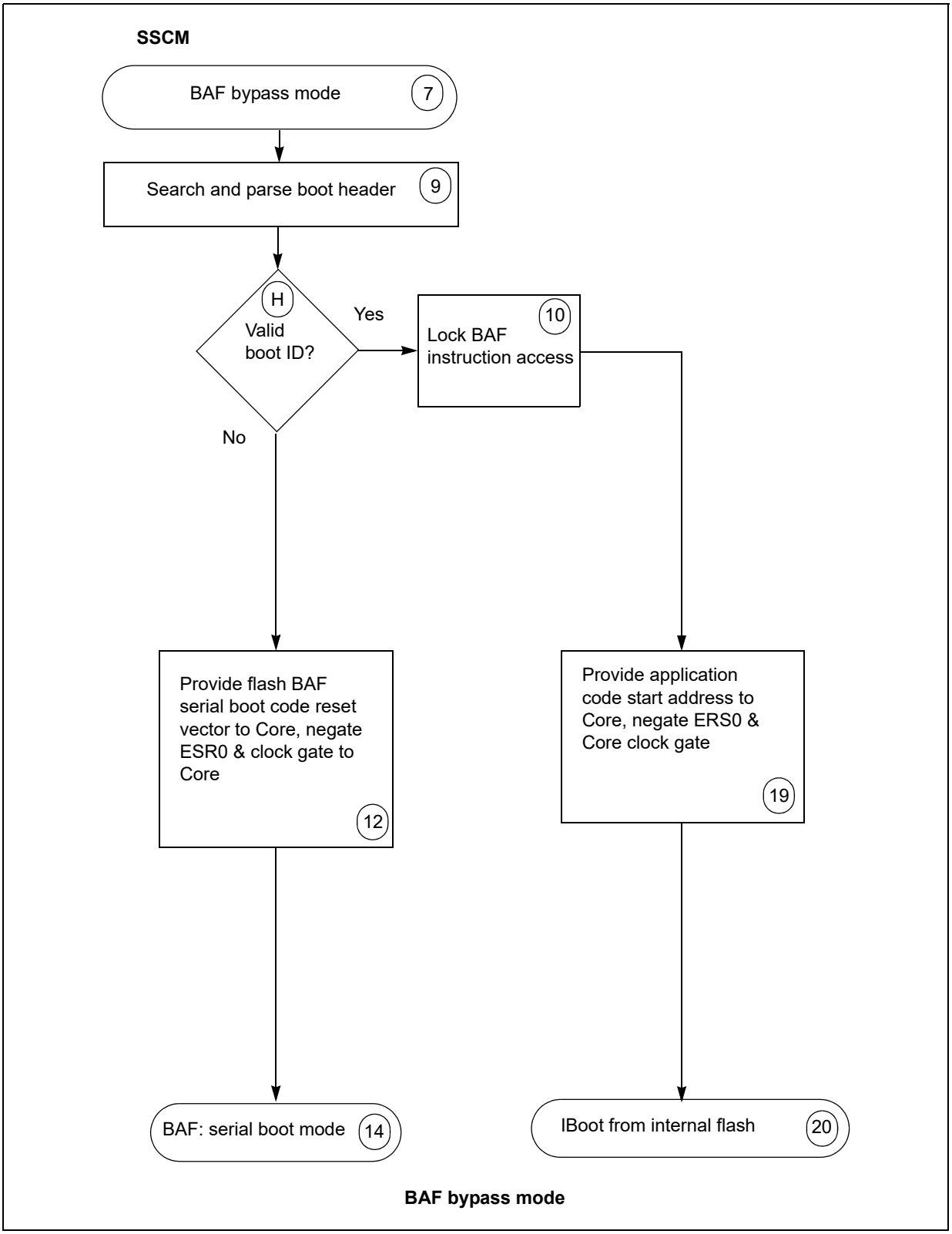
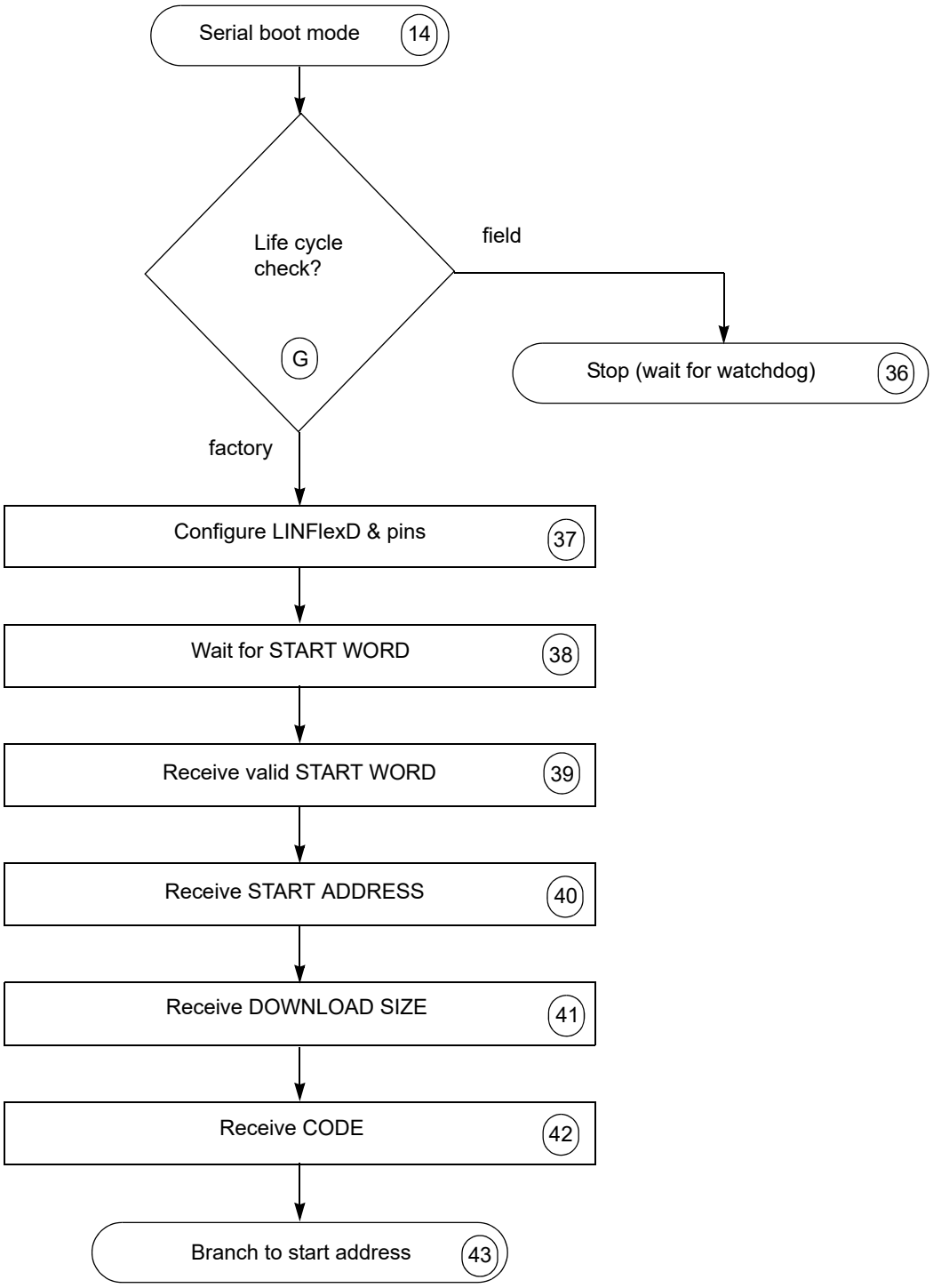


Figure 27. Boot-up sequence, part D



Serial boot process flow diagram

8 Device Configuration Format (DCF) Records

8.1 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 respective data.

There are two broad categories of DCF records:

- TEST DCF records are developed by the factory. They can be used to write any DCF client, but are used mainly to:
  - Program registers involved in trimming trip points for voltage comparators.
  - Adjusting analog to digital voltage supplies.
  - Trim oscillator frequencies.
  - Enable RAM repair.

Note: The TEST DCF Records are programmed into TEST flash during production and cannot be modified. TEST flash is not visible to the user.

- UTEST DCF records can be either:
  - Factory programmed during production testing (see “UTEST flash memory map” table in *Chapter 5: Memory map*).
  - User programmed when application code is written into the flash memory. User-supplied UTEST DCF records start at the next location in the UTEST memory map following the factory UTEST DCF records. The user-defined DCF records set up the initial memory map, assign memory blocks as OTP and 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 CPUs and the DCF records must be properly programmed into their respective flash memories.

When power is applied to a properly programmed device:

1. The Power Management Controller (PMC) takes control of the device until the system power supplies have reached predefined levels.
2. The PMC then signals the Reset Generation Module (RGM) to begin the boot sequence.
3. During boot, the RGM enables the System Status and Control Module (SSCM) to read the device configuration records and write information to specific registers.

8.2 DCF clients

DCF clients are 32-bit 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 like ‘Write Once’ or only allowing bits to be written from ‘1’ to ‘0’ or vice versa. DCF clients need not implement all 32 bits. DCF clients may be designated

'TEST DCF Record Only' so that only DCF Records stored in TEST flash can write to the DCF Client.

A list of DCF clients is shown in *Table 71*. (TEST DCF Record Only clients are not shown as these are not accessible by the user).

The DCF Records select the target DCF client via a 30-bit field in the DCF Record consisting of:

- A 15-bit Chip-Select field—each module that includes DCF clients is assigned a Chip Select during chip definition.
- A 15-bit Address field—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.

8.3 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 28*.

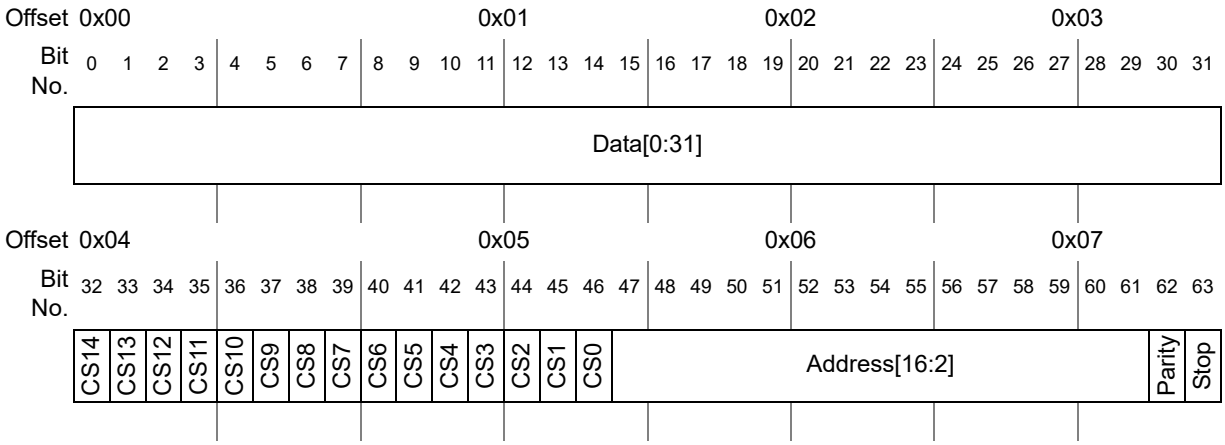


Figure 28. DCF record structure

Table 69. DCF record field description

Field	Description
0–31 Data[0:31]	32 bits of data to be written to the DCF Client.
32–46 CS*n*	Chip Select n. **Note:** Only assert one chip per DCF record to select the target module for the DCF client. All other Chip Selects should be negated. 1 Chip select is asserted 0 Chip Select is negated
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.

Table 69. DCF record field description(Continued)

Field	Description
62 Parity	Parity Bit for the DCF Record. **Note:** This bit is NOT implemented for DCF Clients written from UTEST.
63 Stop	This bit indicates the end of the list of DCF Records. **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. 1 End of the list. 0 NOT the end of 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 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 29* 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 that the following data records must be processed.

Figure 29. DCF start record

0x00 0:31	0x04 32:63
0x05AA55AF	0x00000000

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. Many operational aspects of the device are therefore already configured when the reset signal is released and normal device operation begins.

Some device configuration records are calculated during production testing and programmed into the TEST flash memory area. Other DCF records are developed by the user and programmed into the UTEST flash area.

The stop bit designates the last device configuration record in a set of records. It is not normally set in a valid UTEST DCF record programmed during production as this prevents 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 whose contents is 0xFFFF_FFFF_FFFF_FFFF. The stop bit location in this unprogrammed flash memory location is therefore a logic 1, signifying that this is the last DCF record and not to be acted on.

The general format of the stop record is shown in *Figure 30*. 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 30. DCF stop record

0:31	32:62	63
Reserved	Reserved	1

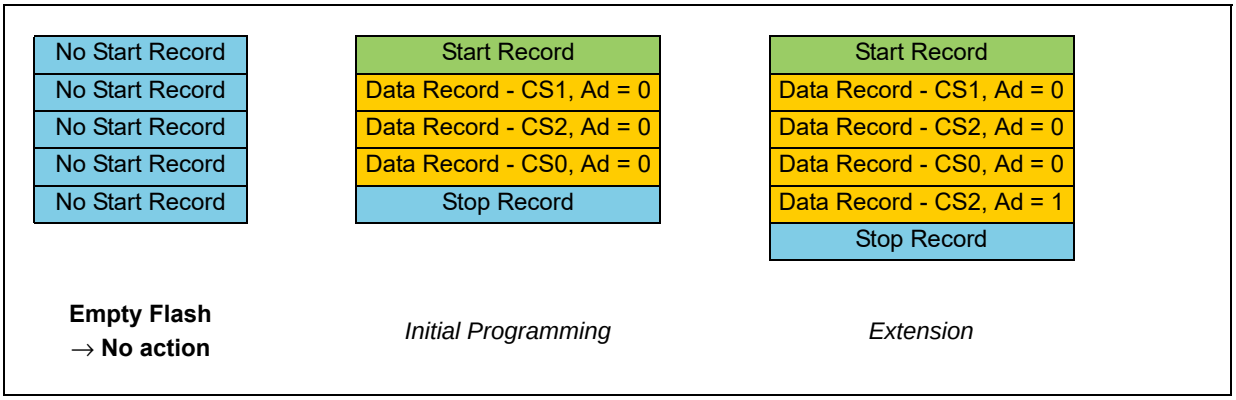
If *n* data records are to be stored in UTEST, the data structure must be as shown in *Table 70*.

Table 70. 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
...	...			
8*n* + 0x0	Reserved			
8*n* + 0x4	Reserved			STOP=1
8(*n*+1) + 0x0	—			
8(*n*+1) + 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. Records to be programmed in several sessions, appending new records to the end of the list each time, is shown in *Figure 31*.

Figure 31. Appending DCF records



It is possible for more than one DCF Record to write to the same DCF client. 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.

8.3.1 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 and the user may add USER DCF records from 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, the records must start at the first address in the UTEST flash memory area and be continuous: 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.

8.4 DCF client table

A list of DCF clients is shown in *Table 71*. Various attributes of the DCF clients are listed in these tables:

- DCF Chip Select—the CS field is a 15-bit field with each bit acting as a module select signal for the internal modules. Only one bit in this field should be set in a DCF record.
- DCF address—the 15-bit ADDR field specifies an internal address for a DCF client. These addresses are only used by the SSCM when interpreting DCF records and writing data to specified modules.
- DCF Client Description—the name of the DCF clients.
- Reset value of DCF client—the reset value of DCF client before being written by the SSCM with the associated DCF record.
- Typical value of DCF client—a typical value that users may program into the various DCF clients. It 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—registers are designated as DCF clients if they need to be written with a DCF record. Other features that must be designated for DCF clients

include different write strategies listed below. The following list defines the special strategies:

– None—no special DCF strategy is used.
– Parity—not implemented for UTEST DCF clients. Used for TEST Only DCF clients not accessible to the user.
– Write Once—a DCF client can only be written once. The DCF client ignores subsequent writes.
– Triple Voted—DCF clients 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 'bit-flip' error tolerance without changing the DCF client output data.
– Triple Voted with second write—as per 'Triple Voted' but during the second execution of Phase 3 of the reset sequence, the SSCM attempts to write the DCF client again and the DCF client checks that the register contains the same data.
– 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 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 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.

8.4.1 DCF client list

Table 71. DCF client list(1)

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
PASS								
000_0000_0000_1000	00000000010 1000	Life Cycle (LCSTAT)	0x00FF	—	yes	Write '0' only	None	8
000_0000_0000_1000	00000000010 1000 – 00000000010 1011	Reserved						
000_0000_0000_1000	00000000010 1100	Censorship	0x55AA	—	yes	None	None	8
000_0000_0000_1000	00000000010 1101 – 00000000010 1111	Reserved						
000_0000_0000_1000	00000000011 0000	Reserved						

Table 71. DCF client list[1](Continued)

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0000_1000	00000000110001 – 00000000111111	Reserved						
000_0000_0000_1000	00000000100000	LOCK0_PG0	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000100001	LOCK1_PG0	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000100010	LOCK2_PG0	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000100011	LOCK3_PG0	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000100100	LOCK0_PG1	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000100101	LOCK1_PG1	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000100110	LOCK2_PG1	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000100111	LOCK3_PG1	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000101000	LOCK0_PG2	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000101001	LOCK1_PG2	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000101010	LOCK2_PG2	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000101011	LOCK3_PG2	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000101100	LOCK0_PG3	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000101101	LOCK1_PG3	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000101110	LOCK2_PG3	0xFFFF_FFFF	—	no	None	None	8
000_0000_0000_1000	00000000101111	LOCK3_PG3	0xFFFF_FFFF	—	no	None	None	8

Table 71. DCF client list[1](Continued)

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_0000_1000	00000000100 1100 – 000000011111 111	Reserved						
OTP								
000_0000_0001_0000	00000000000 0000	Diary Base Address	0xFFFF_FFFC	—	no	Write Once + Write ‘0’ only	None	8
000_0000_0001_0000	00000000000 0001	Tamper Region Override	0x00	—	no	Write Once + Write ‘1’ only	None	8
000_0000_0001_0000	00000000000 0010 – 00000000000 0111	Reserved						
000_0000_0001_0000	00000000000 1000	OTP_EN0	0x0000_0000	—	no	Write ‘1’ only	None	8
000_0000_0001_0000	00000000000 1001	OTP_EN1	0x0000_0000	—	no	Write ‘1’ only	None	8
000_0000_0001_0000	00000000000 1010	OTP_EN2	0x0000_0000	—	no	Write ‘1’ only	None	8
000_0000_0001_0000	00000000000 1011	OTP_EN3	0x0000_0000	—	no	Write ‘1’ only	None	8
000_0000_0001_0000	00000000000 1100 – 00000000001 0011	Reserved						
000_0000_0001_0000	00000000010 1100 – 000000011111 111	Reserved						
000_0000_0010_0000	00000000001 0000 – 000000011111 111	Reserved						
MISCELLANEOUS								

Table 71. DCF client list[1](Continued)

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
000_0000_1000_0000	00000000000 0000	UTEST Miscellaneous (See *Figure 32* and *Table 72* for bit position details.)	0x00000941	—	no	Triple Voted	None	8
000_0000_1000_0000	00000000000 0001 – 11111111111111	Reserved						
BAF ‘Soft’ Clients								
100_0000_0000_0000	00000000000 0000	Security Watchdog Control Register (SEC_SWT_CR)	NA	—	—	Write Once	None	8
100_0000_0000_0000	00000000000 0001	Security Watchdog Timeout (SEC_SWT_TO)	NA	—	—	Write Once	None	8
100_0000_0000_0000	00000000000 0010	Security Watchdog Service Address (SEC_IAC8)	NA	—	—	Write Once	None	8
100_0000_0000_0000	00000000000 0011	Security Watchdog CPU select (SEC_SWT_CPU)	NA	—	—	Write Once	None	8
100_0000_0000_0000	00000000000 0100 – 00000001111111	Reserved						
100_0000_0000_0000	00000010000 0000 – 00000011111111	Reserved						

Table 71. DCF client list[1](Continued)

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description	Reset value of DCF client	Typical value of DCF client	IPS read direct access	DCF client special strategy	DCF order dependency in flash memory	Estimated UTEST flash memory space (bytes)
100_0000_0000_0000	00000100000000	Address (ADDR)	NA	—	—	None	Must be written first as a pair with a data register	8
100_0000_0000_0000	00000100000001	32-bit Data (DATA32)	NA	—	—	None	Must be written second as a pair with the Address register	8
100_0000_0000_0000	00000100000010	16-bit Data (DATA16)	NA	—	—	None	Must be written second as a pair with the Address register	8
100_0000_0000_0000	00000100000011	8-bit Data (DATA8)	NA	—	—	None	Must be written second as a pair with the Address register	8
100_0000_0000_0000	00000100000100 – 00000100001111	Reserved						
100_0000_0000_0000	00000100000100	Callback address	NA	—	—	None	None	8
100_0000_0000_0000	00000100001001 – 00000101111111	Reserved						
100_0000_0000_0000	00000110000000 – 11111111111111	Reserved						

1. DCF programming should start from address 0x00400208.

8.4.2 Miscellaneous DCF registers

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 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	Reserved	XOSC ALC_DIS	XOSC LF_EN	Reserved	Reserved	OSC EN_40MHZ	Reserved	Reserved	XOSC EN	XOSC EXT CLOAD	XOSC LOAD CAP SEL[4:0]					ESR0 GATE
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 32. UTEST Miscellaneous DCF Client

Table 72. UTEST Miscellaneous DCF Client field descriptions

Field	Description
0–16	Reserved
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)
19	Reserved
20	Reserved
21 OSC EN_40MHZ	XOSC EN_40MHz 0 20 MHz 1 40 MHz XOSC
22	Reserved
23	Reserved
24 XOSC EN	XOSC EN — Enable XOSC 0 XOSC disabled 1 XOSC enabled
25 XOSC EXT CLOAD	XOSC EXT CLOAD 0 Selects XOSC internal cap (ok for 20–40 MHz operation) 1 External (required for 4–16 MHz operation)

Table 72. UTEST Miscellaneous DCF Client field descriptions(Continued)

Field	Description
26–30 XOSC LOAD CAP SEL[0:4]	XOSC LOAD CAP SEL [0:4] — Five trim bits to program the load capacitance. Load capacitor values are determined by the crystal manufacturer data sheet. **Note:** To know the capacitance values relative to the 5-bit binary combinations, refer to the "Selectable load capacitance" table of the device datasheet.
31 ESR0 GATE	ESR0 Gate — Gate ESR0 Input 0 ESR0 Input Disabled. ESR0 only asserted as an output from the device. 1 ESR0 Input Enabled. Assertion of the $\overline{\text{ESR0}}$ pin causes a hard system reset.

9 Power management

9.1 Overview

SPC572Lx microcontrollers include a robust power management infrastructure for applications to select various Operational and Low-power modes and monitor internal voltages for High- and Low-voltage conditions. This monitoring is also used to ensure supply voltages and internal voltages are within the required ranges before the microcontroller leaves reset.

This chapter gives an overview of the built-in power management features.

The power management infrastructure encompasses the following modules:

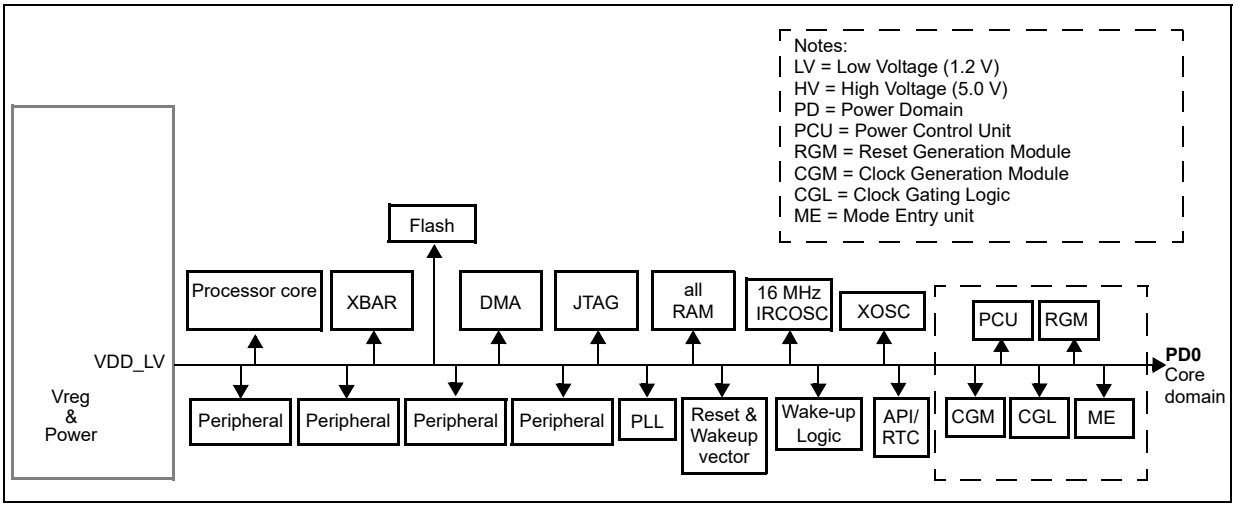
- Power Control Unit (MC_PCU): provides a status register, and the Power Management Control Digital Interface (PMC_dig) submodule, which provides an interface to configure and control internal voltages.
- Clock Generation Module (MC_CGM): generates reference clocks for all the chip blocks. It works in conjunction with internal clock gating logic to implement Low-power modes.
- Mode Entry Module (MC_ME): controls chip operational modes and mode transition sequences. It also contains configuration, control and status registers accessible for the application.
  - System modes RESET, SAFE, TEST and DRUN are entered automatically on system events or on application request.
  - User modes RUNx, HALT, and STOP control performance and consumption during normal application operation. The modes are organized according to the performance provided. Each mode is entered via a dedicated key-protected register to avoid unwanted transition.

See the respective module chapters for further details.

9.1.1 Power management framework

The power management framework supports a variety of configuration options. See *Figure 33* for more details. SPC572Lx implements a unique core voltage power domain. There are no separate functional power domains.

Figure 33. Device power management framework



9.1.2 Power management supply description

Table 73 provides an overview of the PMC supply signals.

Table 73. Device power management controller external signals

Name	Type	Description
VDD_HV_ADR	Reference	Voltage reference of ADC Σ/Δ module Voltage reference of ADC SAR module
VDD_HV_ADV	Supply	High voltage supply for the ADC modules
VDD_HV_IO_MAIN	Supply	High voltage power supply for the I/Os
VDD_HV_IO_ETH	Supply	Ethernet 3.3 V I/O supply
VDD_HV_IO_JTAG	Supply	Oscillator and JTAG pin supply
VDD_HV_PMC	Supply	High voltage power supply for internal power management unit
VDD_LV	Supply	External low voltage power supply for the core area
		Low voltage power supply for the core area generated by internal regulator
VSS	Ground	Ground supply for the device / I/O. This covers both VSS_LV and VSS_HV in case of exposed pad device.
VSS_HV_ADR	Reference	Ground reference of ADC Σ/Δ module Ground reference of ADC SAR module

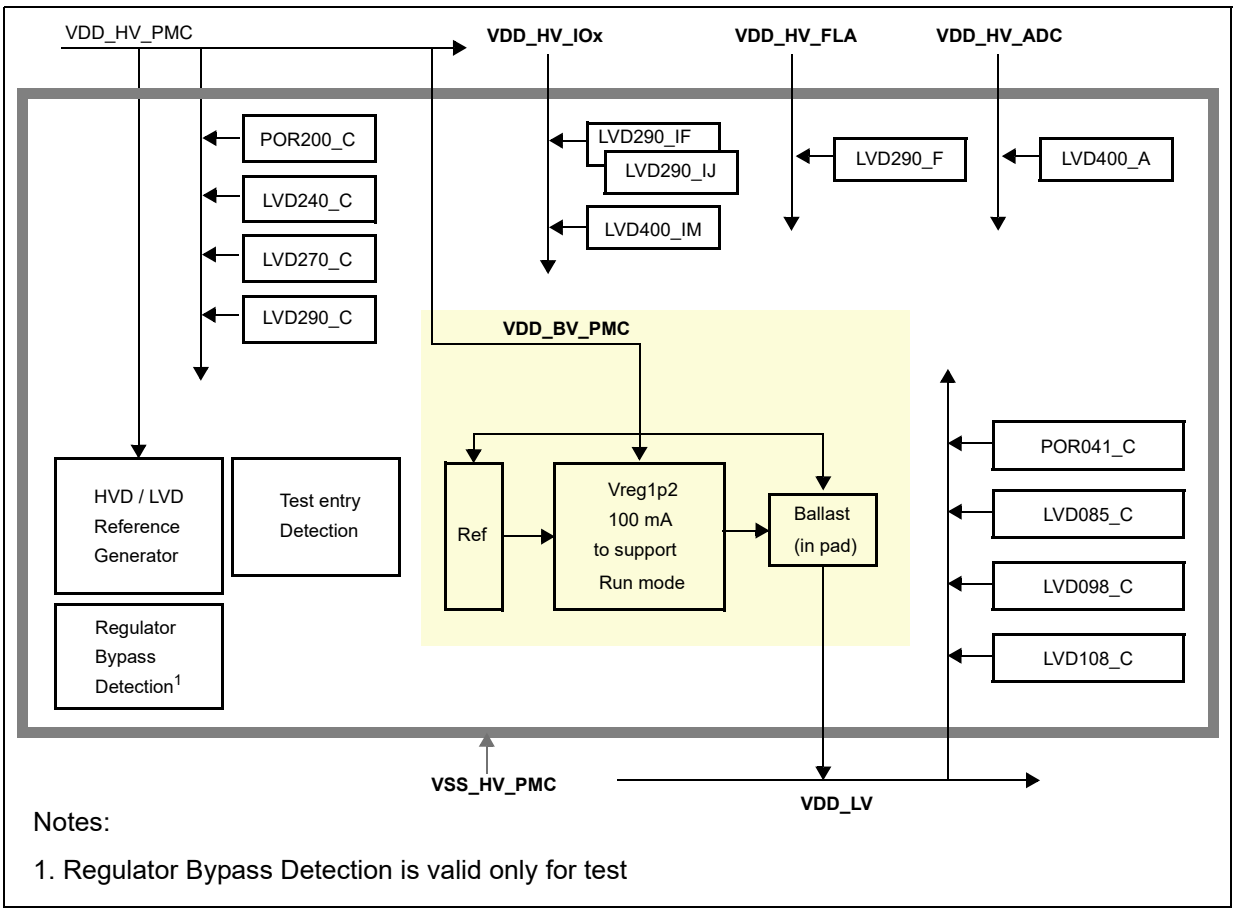
Warning: There are constraints on some input voltages relative to other input voltages. Refer to the device data sheet for detailed information.

9.1.3 SPC572Lx power management controller overview

SPC572Lx power management controller (PMC) implements the following modules:

- Main regulator with internal ballast with dedicated reference.
- LVDs monitoring low voltage supply with characteristics as defined in *Table 74* working on dedicated reference, different from regulator reference.
  - Low voltage detector: LVD085_C, LVD098_C, and LVD108_C
- LVDs monitoring high and medium voltage supplies with characteristics as defined in *Table 74* working on dedicated reference, different from regulator reference.
  - HV POR: POR200_C
  - HV Low voltage detector: LVD240_C, LVD270_C, LVD290_F, LVD290_IF, LVD290_IJ, LVD290_C, LVD400_A, and LVD400_IM
- 2 separate voltage references for safety detection in case of single reference failure:
  - Regulators run on VDD_BV_PMC dedicated supply and dedicated reference.
  - Core supply is monitored using VDD_HV_PMC and associated reference

Figure 34. Device PMC block diagram (simplified)



Notes:

1. Regulator Bypass Detection is valid only for test

9.1.3.1 Core supply main regulator characteristics

Main regulator ensures supply remains in functional range without triggering HVDs/LVDs.

Main regulator uses dedicated reference supplied by VDD_HV_PMC.

9.1.3.2 SPC572Lx power management controller integration

The power management controller has two separate supplies:

- VDD_HV_PMC: provides the main module supply to the power management controller
- VDD_HV_IO_MAIN: provides the low voltage ballast supply.

Note: The internal ballasts are distributed around the device to reduce the temperature gradient on the silicon.

VDD_HV_PMC should always be shorted with VDD_HV_IO_MAIN externally.

Figure 35. Device high voltage power-ground network

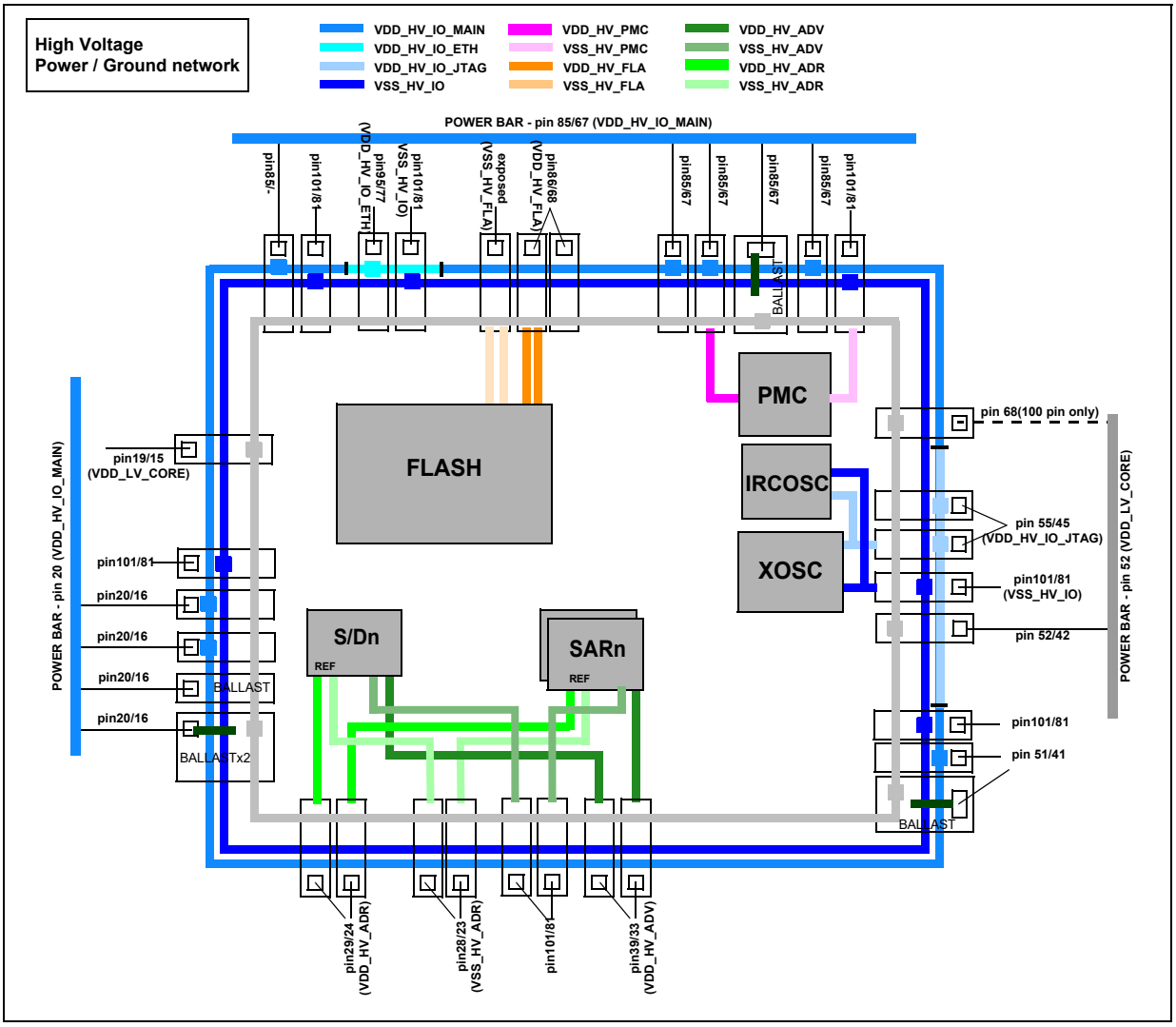
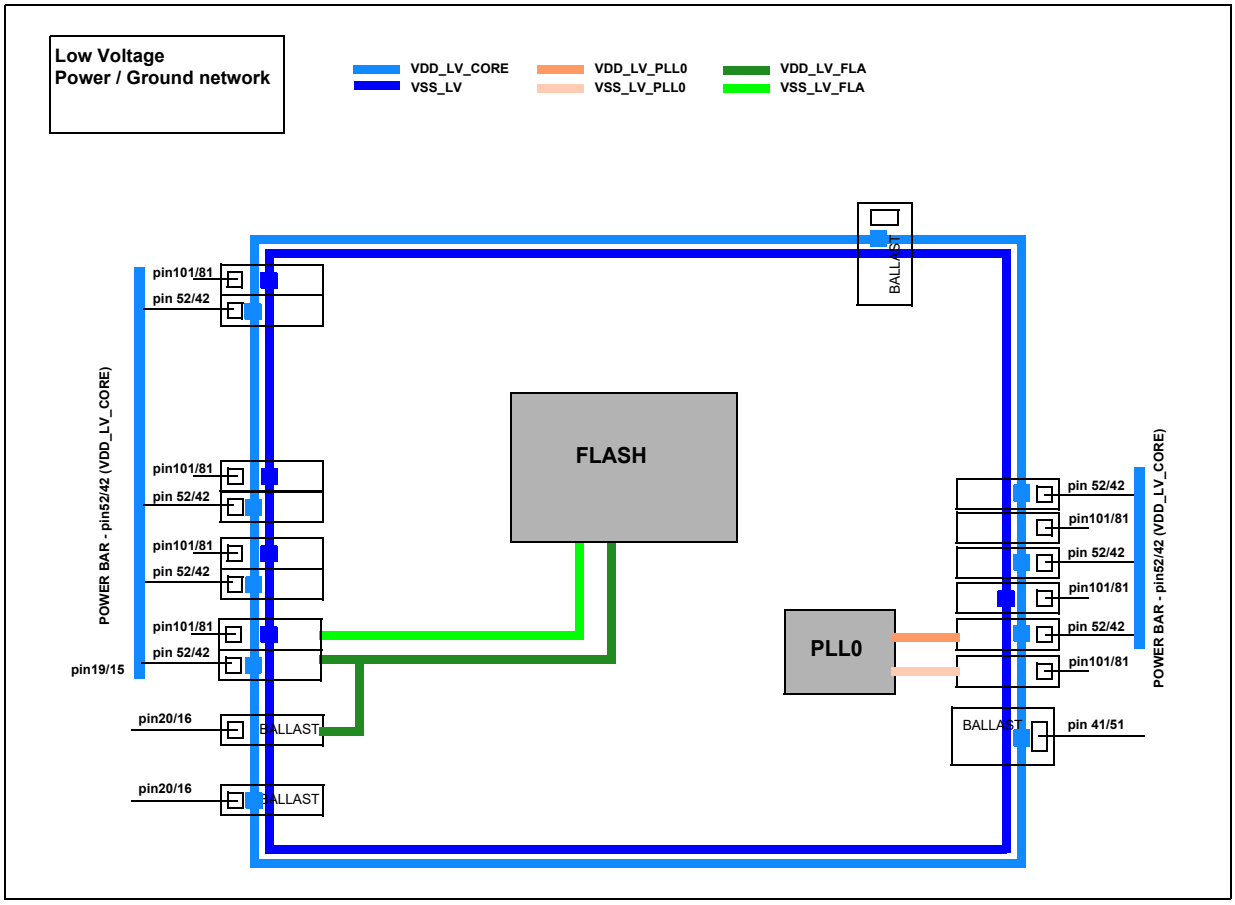


Figure 36. Device low voltage power-ground network



VDD_HV_IO_MAIN redundancy ensures that single pin failure does not prevent device function, although full specification may not be guaranteed.

VDD_HV_PMC redundancy ensures that single pin failure does not prevent device function, although full specification may not be guaranteed.

VDD_LV is redundant, ensuring that single pin failure does not prevent the device from functioning correctly.

The device remains functional when more than four supply pins are available for the same supply, although it may not satisfy all electrical specifications, including EMC performance. This requirement is not applicable to: SAR, S/D, Ethernet, oscillator, JTAG.

9.2 Low power mode support

The SPC572Lx microcontroller supports low power modes.

Dedicated logic prevents uncontrolled activation of low-power mode. This is implemented using protected sequence from the Mode Entry control module MC_ME.

Any register associated with low power entry is triple-voted.

9.2.1 Low power support and STOP mode implementation

Low power mode may be used to control the engine during the power-up phase, as in a Start-Stop system or when the key has been turned off.

After-run is defined as part of RFQ500, the requirement has since been clarified as follows.

The device implements the equivalent of an idle or power-down mode through STOP and HALT mode configuration.

The microcontroller application code can be used to put the device or modules into STOP mode.

A combination of mode control and clock control allows the user to configure the STOP mode.

In After-run mode, peripheral modules can be enabled/disabled in STOP using the control described in the Mode Entry control section. For example, it is possible to make two CAN modules, one ASC module and a few requested pins active. For alternate configurations, other modules can be enabled/disabled in STOP using the control described in the Mode Entry section. External interrupt request channels and external wake-up channels can be active to wake-up the device as alternatives to peripherals interrupts.

All pins associated with the active modules are functional. Control of the pins is defined by user during run-time. A wake-up line is associated with each selectable module.

One of the available modules can be configured to perform the wake-up task. For example, with *n* CAN modules on the chip, one is configured to receive wake-up signals and the others are set to power-down completely. Any of the modules can be configured to do this.

All CAN modules can be independently configured to receive messages and respond to wake-up messages. Control of the pins is defined by user during run-time.

These modules can also be idle in idle mode, but wake-up and wake the core as soon as activity is detected.

All these modules can be configured to a low power state. Wake-up recovery is programmable: for instance, depending on the state or activity of the associated pin.

All parts of the device can be disabled. The following modules are configurable:

- There is one CAN module, selectable from any of the CAN modules on the device. The others can be disabled.
  - The maximum CAN baud rate is 1 MBaud.
  - One CAN module is able to receive messages from the CAN bus and wake-up the controller if the received message contains the correct message identifier.
- One LIN/UART module is able to recognize special wake-up messages on the receive line and then wake up the controller.
- Some request pins are able to initiate a wake-up. Refer to the pinout chapter for more information.
- The system timer (PIT) is able to initiate a wake-up.

Configurable modules (ASC, two CANs, timer) can all be off in STOP mode.

The CAN module recovers from wake-up instantaneously but its ability to capture the message depends on settings of certain peripherals such as the CAN protocol clock. Since the recovery time impacts whether the first message can be captured, software must be used as a workaround.

Device implements all three mechanisms to exit from stop mode: pin wakeup, CAN wake-up and timed wake-up.

Some customer implementations may use an external ASIC regulator that does not have low-power mode support. The Asic regulator is either on or off, with no possibility of changing the regulated output in operation.

9.2.1.1 After-run mode implementation

After-run mode is mainly implemented via clock gating, module disabling, reduced frequency and mode switching. This is done based on revised RFQ500 targets.

In this mode:

- The selected CAN is kept enabled (clocked and active) so it can maintain synchronization with the CAN network.
- The supplied CAN module can buffer the received (first) message and then interrupt the core.
- The core executes from TCM (i.e., Flash is disabled) and uses the internal RC oscillator (IRCOSC) or the external oscillator (XOSC).
- The PMC, LVD, Temperature Sensor, IRCOSC, XOSC and pads are active.
- The PLL, Flash, and ADC are disabled, with rest of the device static and the clock gated off.

9.3 Flash power requirements

During start-up, an untrimmed regulator supplies the flash module. The flash can still be read, but at a slower rate (and reduced voltage requirements).

On completion of the start-up phase, the regulator and the flash module reference current are trimmed so the flash can be read at full specification.

The LV portion of flash is powered directly from the internally generated 1.2 V and the supplied high voltage.

9.4 Device trimming

During the initialization phase, the device defaults to a pre-determined state for each of the LVDs. As the flash becomes available, the differential read process allows the trimming data to be available for the internal LVDs. Refer to *Section 9.6: Power sequence* for further details.

9.5 Supply monitoring (POR and LVDs)

The POR and LVD circuits function correctly even if the input voltage is non monotonic and hold the device in reset until release requisites are met.

9.5.1 Power-on reset (POR)

The PMC implements two internal power-on reset circuits:

- PORUP_LV monitors the voltage on the 1.2 V input VDD_LV_CORE pin and asserts a reset when the input supply is below defined values.
- PORUP_HV monitors the voltage on the 5.0 V input VDD_HV_PMC pin. The POR trip point is set high enough to ensure all the LVD circuits are functional.

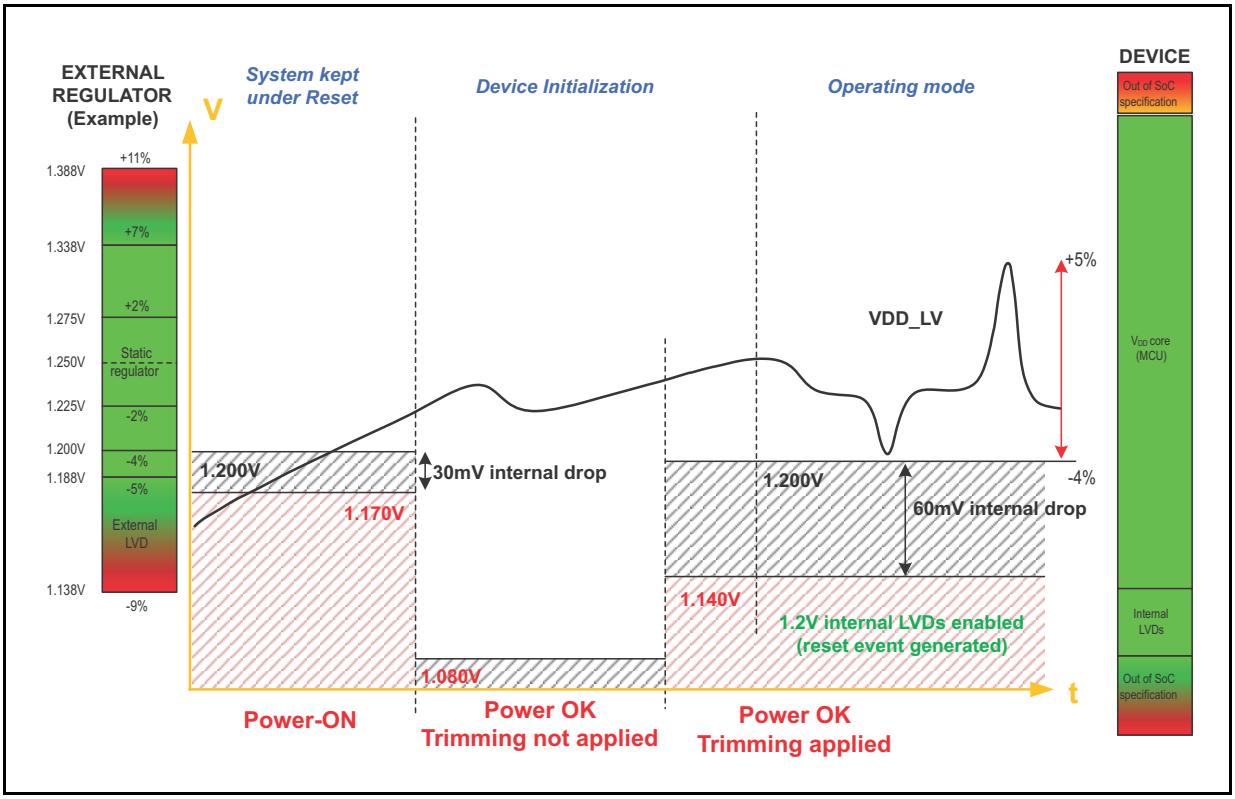
See *Chapter 7: Reset and Boot* for PORST/ESR0 pin functionality.

9.5.2 Behavior of device LVDs

After reset, some LVDs can be disabled (see section *Section 9.5.3: Low Voltage Detection (LVD)*) or used in a 'monitor' only mode, as well as generating a safe/interrupt event. *Figure 37* illustrates the behavior of internal LVD circuits from the power-on phase through to device initialization and subsequent 'normal' run or operating mode.

After initialization, the LVDs can be configured to operate with a higher voltage range by preventing threshold breaches from triggering resets. The application must ensure that the supply remains in the functional range.

Figure 37. Internal LVD and HVD in configuration example during power-up



9.5.3 Low Voltage Detection (LVD)

Internal Voltage Detector circuits monitor the voltage on assigned supplies and assert resets or interrupts when limits are breached. The voltage detectors support hysteresis in the falling and rising trip points.

9.5.3.1 LVD implementation

- All LVDs are capable of generating an RGM event, with the following exceptions:
  - LVD270 monitor high voltage supply input and always generate a reset when triggered (can only be disabled for test purposes).
  - LVD098 monitor the internal core voltage and always generate a reset when triggered (cannot be disabled). LVD098 is integral to POR management.
- All LVDs configured for reset generation can cause functional or destructive resets. MC_RGM cannot exit PHASE0 until all destructive reset conditions are cleared.
- The appropriate bits in the PMC registers are set by LVD events.
- Modification of LVD and HVD control registers is subject to procedures in *Chapter 67: Register Protection (REG_PROT)*.
- REE DCF records configure LVDs/HVDs to a RESET event. This is a write-once mechanism managed by SSCM during device initialization. When a REE DCF record configures a RESET event, it cannot be changed until the next RESET event.

Note: Details regarding configurable LVDs/HVDs are provided in Table 74

- The external PORST pin is driven low when a destructive-reset-enabled LVD is triggered.

Note: LVD098 and LVD270 (low range LVDs) cannot be disabled by the user during boot sequence. These modules are used during the power-up phase and ensure that the absolute lowest threshold of operation is not breached. This is not a guarantee that the device will function down to this level, only that the device will recover if this level is crossed.

Warning: LVD270 monitors can be disabled for test purposes via software access of the PMC_REE register. However, the internal monitors of the device may not detect that VDD_HV voltage is outside functional range (below LVD270 threshold) so the user must ensure a reset state by assertion of the external PORST pin.

Note: When supply falls below the internal power-on state detector threshold, the device automatically resets and LVD270 is re-enabled.

Table 74. POR and voltage monitors description

Monitor name	Register bit	Description	Monitor/ sensing	POWERUP-PHASE0	PHASE1-2-3	User configurability	
						DCF (PHASE3)	REE/IE(1) (RUN)
POWER MANAGEMENT UNIT VOLTAGE MONITORS – LOW VOLTAGE SUPPLY							
LVD098_C	VD2	– Low voltage core supply – low voltage detector	VDD_LV_CORE	ENABLE	ENABLE	ENABLE	ENABLE
LVD108_C	VD3	– Low voltage core supply – low voltage detector	VDD_LV_CORE	DISABLE	DISABLE	LVDM[0]	VD3_C
POWER MANAGEMENT UNIT VOLTAGE MONITORS – HIGH VOLTAGE SUPPLY							

Table 74. POR and voltage monitors description(Continued)

Monitor name	Register bit	Description	Monitor/ sensing	POWERUP-PHASE0	PHASE1-2-3	User configurability	
						DCF (PHASE3)	REE/IE(1) (RUN)
LVD270_C	VD9	– High voltage core supply – low voltage detector	VDD_HV_PMC	ENABLE	ENABLE	ENABLE	ENABLE
LVD290_C	VD10	– High voltage core supply – low voltage detector	VDD_HV_PMC	ENABLE	ENABLE	LVDM[14]	VD10_C
LVD290_F	VD10	– High voltage flash supply – low voltage detector	VDD_HV_FLA	ENABLE	ENABLE	LVDM[15]	VD10_F
LVD290_IF	VD10	– High voltage Ethernet supply – low voltage detector	VDD_HV_ETH	ENABLE	ENABLE	LVDM[16]	VD10_IF
LVD290_IJ	VD10	– High voltage JTAG supply – low voltage detector	VDD_HV_JTAG	ENABLE	ENABLE	LVDM[17]	VD10_IJ
LVD400_A	VD14	– High voltage ADC supply – low voltage detector	VDD_HV_ADV	ENABLE	DISABLE	LVDM[8]	VD14_A
LVD400_IM	VD14	– High voltage I/O main supply – low voltage detector	VDD_HV_IO_MAIN	ENABLE	DISABLE	LVDM[7]	VD14_IM

1. REE: Reset event enable. IE: Interrupt enable — See *Chapter 54: Power Management Controller digital interface (PMC_dig)*.

Table 75 provides the monitor status depending on configuration.

Table 75. Voltage monitors configurability

POWER UP	PHASE 0-1-2	DCF	REE(1)	RES(2)	IE(3)	Monitor status
ENABLE	ENABLE	—	—	—	—	Default configuration (boot threshold monitor) – Monitor full run from power-up to power-down
ENABLE	DISABLE	ENABLE	ENABLE	DEST	—	– Monitor power-up, gate exit from power-up phase. – Monitor disabled on phase0 entry. – Monitor re-enabled in PHASE 3 on DCF rec reading. – Trigger destructive event reset during reset flow. – Configuration is locked to enable during run time.

Table 75. Voltage monitors configurability(Continued)

POWER UP	PHASE 0-1-2	DCF	REE(1)	RES(2)	IE(3)	Monitor status
ENABLE	DISABLE	ENABLE	ENABLE	FUNC	—	– Monitor power-up, gate exit from power-up phase. – Monitor disabled on phase0 entry. – Monitor re-enabled in PHASE 3 on DCF rec reading. – Trigger destructive event reset during reset flow. – Configuration is locked to enable during run time. After reset completion: – LVD event demoted from destructive to functional reset by software after reset sequence completion.
ENABLE	DISABLE	DISABLE	DISABLE	—	ENABLE	– Monitor power-up, gate exit from power-up phase. – Monitor disabled on phase0 entry. – Monitor re-enabled in PHASE 3 on DCF rec reading. After reset completion:(4) – LVD event is also associated to an interrupt event.
ENABLE	DISABLE	DISABLE	DISABLE	—	DISABLE	Default configuration (hysteresis monitor). – Monitor power-up, gate exit from power-up phase. – Monitor disabled on phase0 entry. – Monitor re-enabled in PHASE 3 on DCF rec reading. After reset completion:(4) – see footnote
DISABLE	DISABLE	ENABLE	ENABLE	DEST	—	– Monitor enabled in PHASE 3 on DCF record reading. – Trigger reset event. – Configuration locked to enable during run time.
DISABLE	DISABLE	ENABLE	ENABLE	FUNC	—	– Monitor enabled in PHASE 3 on DCF record reading. – Trigger reset event. – Configuration locked to enable during run time. After reset completion: – LVD event demoted from destructive to functional reset by software after reset sequence completion.

Table 75. Voltage monitors configurability(Continued)

POWER UP	PHASE 0-1-2	DCF	REE(1)	RES(2)	IE(3)	Monitor status
DISABLE	DISABLE	DISABLE	DISABLE	—	ENABLE	– Monitor enabled in PHASE 3 on DCF record reading. After reset completion:(4) – LVD event is also associated to an interrupt event.
DISABLE	DISABLE	DISABLE	DISABLE	—	DISABLE	Default configuration (application monitor). – Monitor enabled in PHASE 3 on DCF record reading. After reset completion:(4) – see footnote

1. REE: Reset event enable
2. RES: Reset event select
3. IE: Interrupt enable
4. Configuration can be changed by software to any of following events: functional reset, destructive reset or interrupt event by software

When using an external regulator for the 1.2 V generation, the voltage regulator system shall have an internal LVD to satisfy the safety requirement.

An external 4.5V LVD will always be present in the system and hence there is no requirement for one on the device.

9.6 Power sequence

The following sections describe the power sequence and the relation between the different supplies during power-up and power-down.

An internal power-on signal identifies the POWERUP state when the device is not supplied or partially supplied, and is released high on power-sequence exit. The signal is a combination of LVDs monitoring:

- VDD_LV
- VDD_HV_PMC
- VDD_HV_IO
- VDD_HV_FLA

LVD thresholds are configurable by hardware (flash option bits content) or by software (LVD event configuration via register interface). Once the power-on signal is asserted, device configuration resets to default power-up configuration.

9.6.1 Power-up sequence

In this section it is assumed that all supplies are low when entering the power-up sequence. Brown-out and power-down sequences are discussed separately.

9.6.1.1 POWERUP state

At the beginning of power-up, the internal power-on signal remains low due to internal parasitic diodes. When the minimum threshold is reached on VDD_LV and VDD_HV supply, the internal power-on signal is forced low until power-up LVDs reach upper un-trimmed threshold (refer to datasheet).

The different supplies can rise independently as long as the constraints described in the datasheet are met.

During power-up, all functional terminals are maintained in a known state as described in *Table 76*.

Table 76. Functional terminal state during power-up and reset

Terminal type(1)	POWERUP(2) pad state	RESET pad state	DEFAULT pad state(3)	Comments
PORST	strong pull-down	weak pull-down	weak pull-down	Power-on reset pad
ESR0	strong pull-down	strong pull-down	weak pull-up(4)	Functional reset pad
ESR1	weak pull-up	weak pull-up	weak pull-up	Functional NMI/reset pad
TEST_MODE	weak pull-down	weak pull-down	weak pull-down	—
GPIO	weak pull-up	weak pull-up	weak pull-up	—
ANALOG	high impedance	high impedance	high impedance	—
ERROR	high impedance	high impedance	high impedance	
JCOMP(5)	high impedance	weak pull-down	weak pull-down	—
TCK(5)	high impedance	weak pull-down	weak pull-down	—
TMS(5)	weak pull-up	weak pull-up	weak pull-up	—
TDI(5)	weak pull-up	weak pull-up	weak pull-up	—
TDO	high impedance	high impedance	high impedance	—

1. Refer to pinout information for terminal type
2. POWERUP state is maintained until supply cross the power-on reset threshold: V_{PORUP_LV} for LV supply, V_{PORUP_HV} for high voltage supply.
3. Before software configuration
4. ESR0 configuration moves from strong pull-down to weak pull-up on completion of internal reset sequence (PHASE3[FUNC]).
5. JTAG interface implements built-in pull-up/pull-down configuration to ensure noise immunity. Worst case is boundary scan activation, requiring JCOMP to pulse high and following events to occur during JCOMP pulse high:
   - 23 TCK events corresponding to12 clock cycles.
   - 3 TMS events synchronous with TCK: sequence 0,1,1,0,0,0,0,0,0,0,0,1
   - 2 TDI events synchronous with TCK: sequence x,x,x,x,x,0,0,0,1,0,0,x

   TCK, TMS, TDI are implementing hysteresis requiring minimum variation

Digital reset is asserted, ensuring that all module registers are reset to power-on value.

9.6.1.2 POWERUP exit

The device exits POWERUP when VDD_LV and VDD_HV are above the internal LVD thresholds.

9.6.1.2.1 VDD_LV conditions and LVD trimming/enabling sequence

During POWERUP, the enabled LVDs are:

- LVD098_C

VDD_LV exits POWERUP when:

- LVD098_C upper threshold is crossed

1. Once the POWERUP exit conditions (low and high voltage) have been verified, the internal power-on signal is released to all analog modules.
2. The internal RC oscillator module (IRCOSC) starts initialization and provides clock to the system after $t_{RCSTARTUP}$. The PMC digital interface reset is released after two RC clock cycles.

   LVD096_C, and LVD112_C share the same reference. LVD112_C – LVD096_C > 90 mV ± 5%. This difference provides margin with respect to maximum internal resistive drop (40 mV at I_{max}) and 50 mV hysteresis addressing external supply drop.
3. The device proceeds through the RGM reset sequence: PHASE0–PHASE1–PHASE2–PHASE3.
4. Voltage detector (LVD) modules are trimmed at the beginning of PHASE3.
   - Trim values for LVDs are stored as internal DCF records.
   - The SSCM uses the flash low-voltage (low-speed) read feature to access the DCF records and apply the trim values to each LVD module.
   - After analog delay ($t_{LVDTRIM}$) has elapsed, the PMC acknowledges that all LVDs have been trimmed and supply is above threshold
   - the SSCM proceeds with the reset sequence, eventually accessing the flash at full speed to read the option bits required to complete device configuration.
5. The configurable LVD modules can optionally be unmasked at the end of PHASE3. Mask information is read as DCF record from the flash UTEST option bits.

When LVD108_C is masked by the application using the flash user option bits, the device should rely on PORST signal to detect a voltage failure during power-up: PORST must be released high before proceeding with the power-up sequence, which may increase reset sequence completion time.

Figure 38 provides an example of VDD_LV power-up sequence from POWERUP to PHASE3.

Figure 38. Example of VDD_LV power-up sequencing

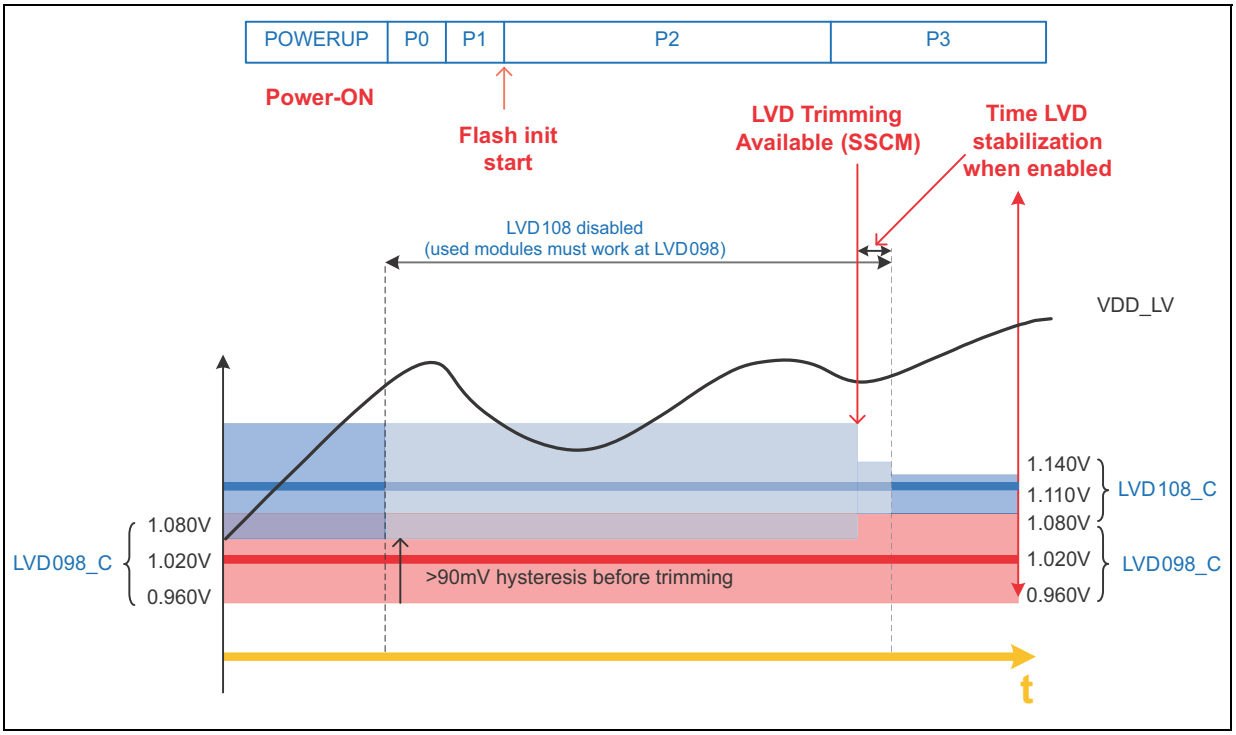
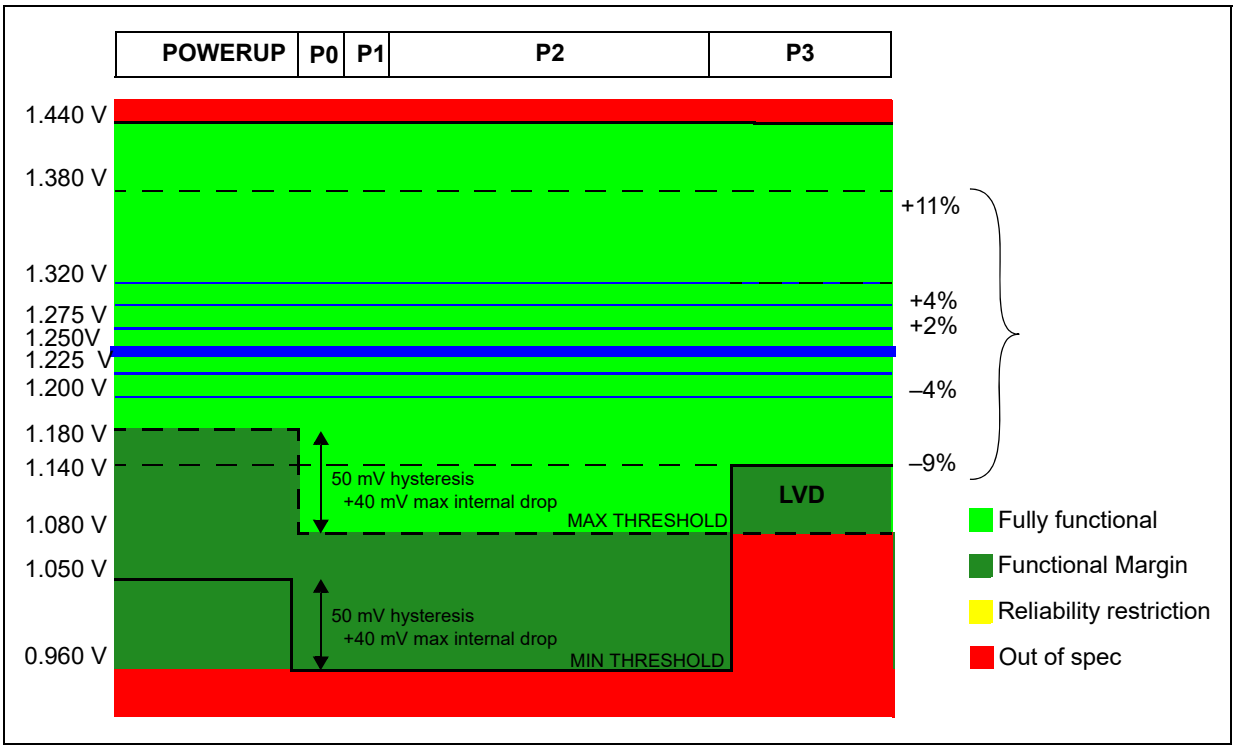


Figure 39 illustrates threshold variation of VDD_LV LVD monitor during power-up sequence from POWERUP to PHASE3.

Figure 39. Threshold variation during power-up sequence



9.6.1.2.2 VDD_HV conditions and LVD trimming/enabling sequence

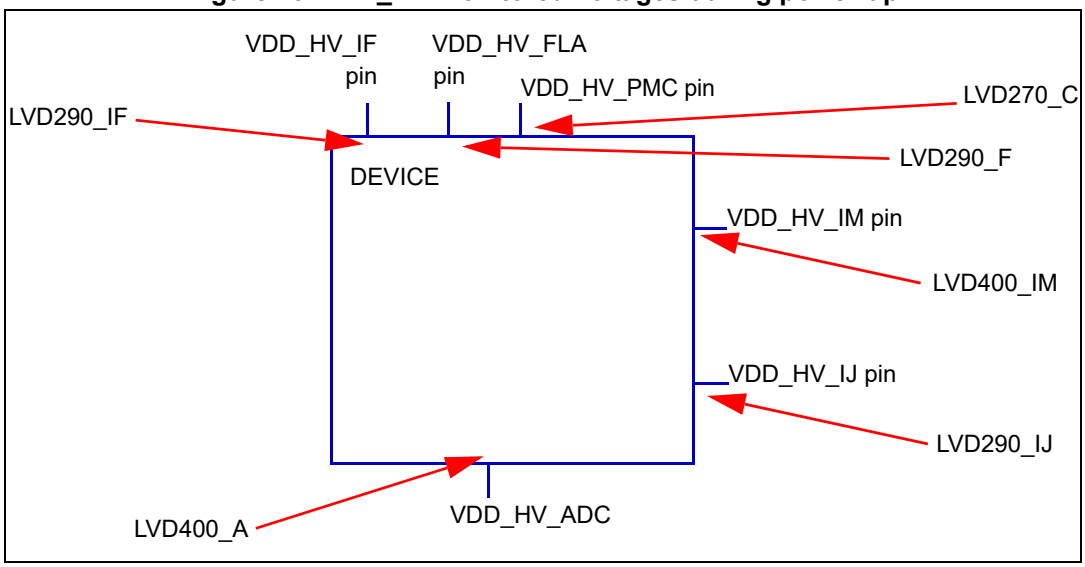
During POWERUP, the following high voltage LVDs are enabled:

- LVD400_IM
- LVD270

The VDD_HV conditions to exit POWERUP are the following:

- LVD400_IM upper threshold is crossed.
- LVD290_F upper threshold is crossed.
- LVD270 upper threshold is crossed.

Figure 40. VDD_HV monitored voltages during power-up



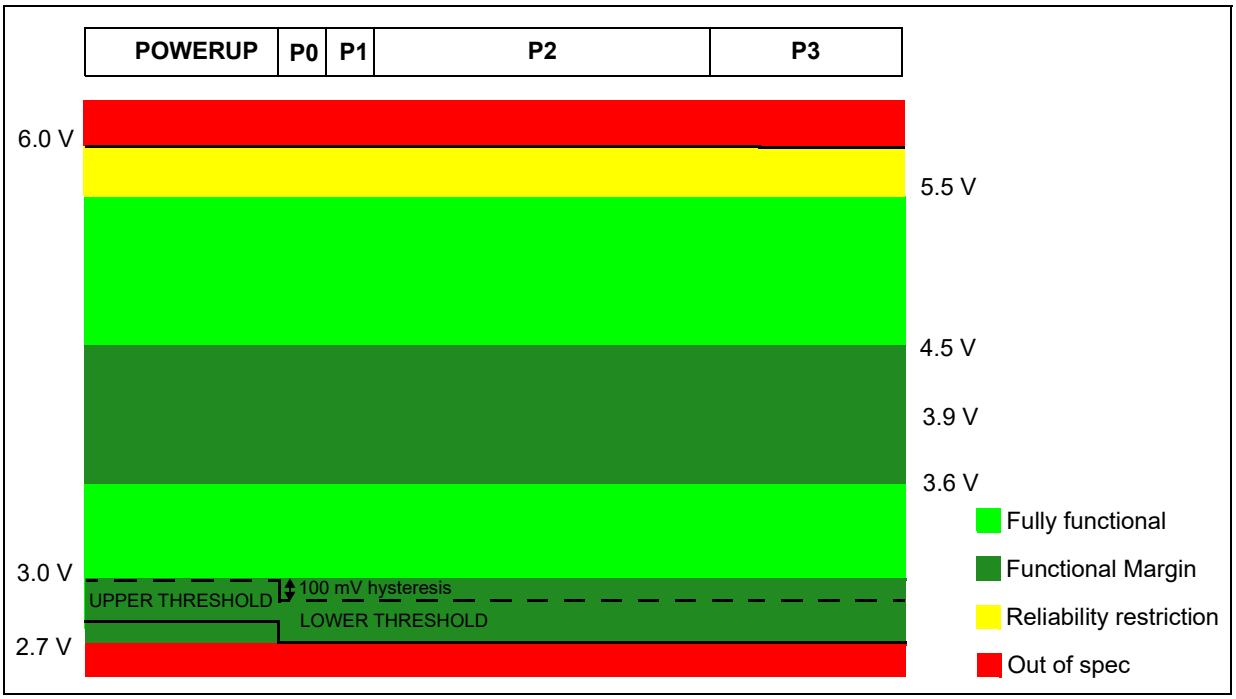
1. After both LV and HV POWERUP exit conditions have been verified, ipp_por_en_b signal is released to all analog modules.
2. The IRCOSC module starts initialization and provides the clock to the system after $t_{RCSTARTUP}$. PMC digital interface reset is released after two RC clock cycles and LVD400_IM is masked. All other HV LVDs remain active.

   LVD270_C and LVD400_IM share the same reference. The LVD400_IM - LVD270_C difference represents the margin with respect to the maximum internal resistive drop and hysteresis addressing external supply drop.
3. The device proceeds through the RGM reset sequence: PHASE0–PHASE1–PHASE2–PHASE3.
4. Voltage detector (LVD) modules are trimmed at the beginning of PHASE3.
   - Trim values for LVDs are stored as internal DCF records.
   - The SSCM uses the flash low-voltage (low-speed) read feature to access the DCF records and apply the trim values to each LVD module.
   - After analog delay ($t_{LVDTRIM}$) has elapsed, the PMC acknowledges that all LVDs have been trimmed and supply is above threshold
   - the SSCM proceeds with the reset sequence, eventually accessing the flash at full speed to read the option bits required to complete device configuration.
5. The configurable LVD modules can optionally be unmasked at the end of PHASE3. Mask information is read as DCF record from the flash UTEST option bits.

Figure 41 provides the threshold variation of VDD_HV LVDs monitor during power-up sequence from POWERUP to PHASE3.

Internal LVD270_C is enabled during PHASE3.

Figure 41. Threshold variation during power-up sequence



The device needs to fulfill all conditions described on *Section 9.6.1: Power-up sequence* to recover.

9.6.2 Power-down sequence

When the threshold of the higher-range LVDs configured to generate a destructive reset is crossed but neither LVD098 nor LVD270 threshold are crossed, the device enters PHASE0.

1. The power-down sequence initiates when the LVD098 or LVD270 threshold is crossed
2. The device enters the POWERUP state and the internal power-on signal ipp_por_en_b is asserted.
3. On ipp_por_en_b assertion, all analog modules reach their safe state.
4. Device supplies may then proceed to drop down to ground either through device leakage or external pull-down. During power-down, all supplies should comply with supply constraints.

9.6.3 Brown-out management

During brown-out, the device re-enters the POWERUP phase as soon as the threshold of either LVD098 or LVD270 is crossed. The internal power-on signal ipp_por_en_b is asserted.

Note: The device will correctly start independently from any residual voltage.
The device remains in the POWERUP state until the voltage rises above the LVD270

threshold. It then passes to a RESET state defined by Voltage monitor or external PORST pin, or ACTIVE state when all have been released.

Warning: LVD270 monitors can be disabled for test purposes via software access of the PMC_REE register. However, the internal monitors of the device may not detect that VDD_HV voltage is outside functional range (below LVD270 threshold) so the user must ensure a reset state by assertion of the external PORST pin.

Note: When supply falls below the internal power-on state detector threshold, the device automatically resets and LVD270 is re-enabled.

On internal power-on signal assertion, all analog modules reach their safe state.

9.6.4 Low voltage requirement during crank

The device can continue operation to the minimum input voltage during crank.

In order to proceed with execution during cranking and prevent device reset, it is important to correctly configure the high voltage LVDs.

During device switch-off, external LVD monitoring may not be activated quickly enough (external assertion of the PORST pin) to avoid the –10% level being crossed before POR is asserted. Malfunction is possible in this case, but reliability is not impacted and flash is not corrupted.

Internal LVDs that monitor the supply ensure that the flash content is protected.

10 Security

This device has a comprehensive set of customer-configurable security features designed to protect code and data from unauthorized access.

10.1 Basic security

All SPC572Lx 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 SPC572Lx 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
- Secure watchdog timer
- Basic debugger restrictions (on/off via censorship mode)

10.2 Advanced security

Depending on the device configuration, the following features are available:

- Secure debugger interface
- Boot mode:
  - Trusted/secure boot support

10.3 Detailed security information

Details of most of the SPC572Lx security features are published in a separate *SPC572Lx Microcontroller Security Reference Manual*, which is only available to qualified customers.

11 Calibration and Debug

The chapter discusses the device specific information for the debug and calibration modules. The device starts up in JTAG mode. All standards used have freely available specifications for tool developers.

The Event Out ($\overline{\text{EVTO}}$[0,1]) signals are connected to the external TGOUT[0,1] device pins and the Event In ($\overline{\text{EVTI}}$) signals are connected to the external TGIN[0,1] device pins.

11.1 Core debug support

Internal debug support in the e200z215An3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. The core has the capability of exiting reset halted in debug mode, such that no code runs. This allows debuggers to perform operations such as configuration of calibration remap or edit of overlay memory prior to user code execution.

For details, see *Chapter 57: e200z215An3 Core Debug Support*.

11.2 Run control and memory access

Run control is any operation required to control the device operation. Run control includes starting and stopping execution of code by the processor, as well as access to memory while the device is stopped to download code. The Debug and Calibration Interface (DCI) provides the central run control including cross-triggering of breakpoint conditions. This DCI provides support for the IEEE 1149.1 JTAG standard with the JTAG Controller (JTAGC) as well as the IEEE 1149.7 standard with the Compact JTAG (CJTAG) module. The JTAG interface supports both boundary scan and debug modes. It is based on the IEEE 1149.1 standard and the IEEE 1149.7 standards. The JTAG Controller (JTAGC) operates in the standard IEEE 1149.1 5-wire interface. The JTAGC can be fed either directly from the JTAG pins or from the CJTAG module, which supports the IEEE 1149.7 standard.

The Sequence Processing Unit (SPU) provides cross triggering of events from the device. The SPU can access some of the DCU run control features since it can control the $\overline{\text{EVTO}}$ pins and monitor the $\overline{\text{EVTI}}$ pins. Further, the SPU can use system triggers to cause many different types of actions, including triggering a breakpoint cross-trigger (managed by the DCI).

Table 77. Reference links to related information

Related module	Reference
Debug and Calibration Interface (DCI)	*Chapter 58: Debug and Calibration Interface (DCI)*
JTAG Controller (JTAGC)	*Chapter 59: JTAG Controller (JTAGC)*
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)	*Chapter 60: IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)*
JTAG Data Communication (JDC)	*Chapter 61: JTAG Data Communication (JDC)*
Sequence Processing Unit (SPU)	*Chapter 63: Sequence Processing Unit (SPU)*

11.2.1 Debug and Calibration Interface (DCI)

The Debug and Calibration Interface (DCI) module provides debug and calibration features. The DCI module includes the device JTAG controller, the IEEE 1149.7 interface. The DCI provides the following features:

- Debug mode enable control for connected tool or software
- IEEE 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for CPU when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug and calibration features may operate simultaneously

The DCI sends and receives the $\overline{\text{EVTI}}$[1:0] and $\overline{\text{EVTO}}$[1:0] signals to/from the core and other modules. *Table 78* summarizes the sources and destinations of $\overline{\text{EVTI}}$, $\overline{\text{EVTO}}$, and relative signals for the core. For example, $\overline{\text{EVTI}}$[0] are outputs of the JTAGM and SPU, and inputs to the core and GTM.

Table 78. DCI signals

Signal	Core	GTM	NAR	JTAGM	SPU	DTS
$\overline{\text{EVTI}}$[0]	I	I	—	O	I/O	—
$\overline{\text{EVTI}}$[1]	—	—	—	O	I/O	—
$\overline{\text{EVTO}}$[0]	O	O	—	I	I/O	—
$\overline{\text{EVTO}}$[1]	—	—	—	I	I/O	O
Core debug event	I	—	—	—	—	—
DCI cross triggering	O	O	—	—	—	—

The DCI sends the system debug signal, ipg_debug, to the following modules. For details regarding module functionality during system debug, refer to the individual module chapter.

- CAN
- LFAST (Interprocessor Communications instance only)
- DSPI
- eDMA
- GTMINT
- PIT
- SARADC
- SDADC
- SRX
- SSCM
- STM
- SWT

11.2.1.1 DCI $\overline{\text{EVTx}}$ pin multiplexing control register (DCI_PINCR)

The DCI $\overline{\text{EVTx}}$ pin multiplexing control register (DCI_PINCR) is described below. The $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ functions share the same set of possible pins. If this register is written to enable both $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ on the same pin, the $\overline{\text{EVTI}}$ function is selected, although this type of programming should be avoided. Programming the $\overline{\text{EVTI}}$ fields to select $\overline{\text{EVTI}}$[0] or $\overline{\text{EVTI}}$[1] inputs to come from more than one pin is considered a programming error and no pin is enabled.

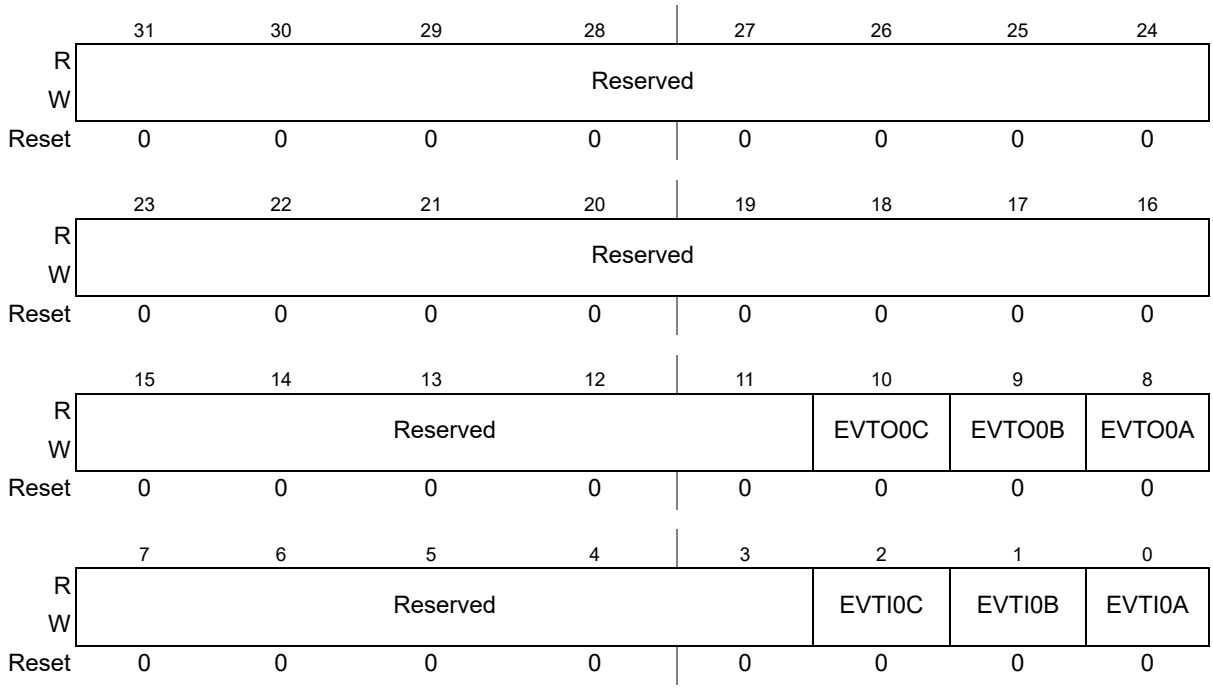


Figure 42. DCI EVTx pin multiplexing control register (DCI_PINCR)

Table 79. DCI_PINCR register field descriptions

Field	Description
31–11	Reserved
10 EVTO0C	Enables $\overline{\text{EVTO}}$[0] function on pin PD[6]
9 EVTO0B	Enables $\overline{\text{EVTO}}$[0] function on pin PA[9]
8 EVTO0A	Enables $\overline{\text{EVTO}}$[0] function on pin PA[8]
3–7	Reserved
2 EVTI0C	Enables $\overline{\text{EVTI}}$[0] function on pin PD[6]

Table 79. DCI_PINCR register field descriptions(Continued)

Field	Description
1 EVTI0B	Enables $\overline{\text{EVTI}}$[0] function on pin PA[9]
0 EVTI0A	Enables $\overline{\text{EVTI}}$[0] function on pin PA[8]

11.2.1.2 JTAG Controller (JTAGC)

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. All Nexus debug registers are accessed and configured via the JTAG controller.

11.2.2 JTAG Data Communication (JDC)

The JTAG Data Communication (JDC) module allows access to two registers by JTAG and software running on the CPU. These registers can be used to exchange data between the internal CPU and an external debug tool. These registers can be used by security applications to exchange passwords or to implement a challenge/response authorization protocol.

11.2.3 Sequence Processing Unit (SPU)

The Sequence Processing Unit (SPU) provides on-device logic analyzer trigger functions, such as performance monitoring, using internal device signals as trigger sources. 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 allow counting and timing of any debug trigger combinations supported by the SPU.

Various clients generate watchpoints and other triggers when operating in their 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, complex logic-analyzer-like behavior can provide vital real-time visibility and debugging of the activities of the system.

Note: SPU literature in this reference manual may refer to the SPC572Lx core via “CPU2”. This is associated with the CPU naming convention applied across the family to which this device pertains and always refers to the single core present on it.

11.2.3.1 SPU Level1 input Mux configurations registers

The SPU Level1 Mux inputs for the L1SELn registers (where n = 0–7) are described in the following tables.*Table 80* contains the L1SEL0 register field descriptions.

Table 80. L1SEL0 register field descriptions

Field	Description
MUX 0	000 Reserved 001 Reserved 010 CPU instruction address compare 1 (watchpoint 0) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 1	000 Reserved 001 Reserved 010 CPU instruction address compare 2 (watchpoint 1) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 2	000 Reserved 001 Reserved 010 CPU instruction address compare 3 (watchpoint 2) 011 Reserved 100 Reserved 101 Reserved 110 CPU debug event input 1 (watchpoint 10) 111 CPU debug event output 2 (watchpoint 20)
MUX 3	000 Reserved 001 Reserved 010 CPU instruction address compare 4 (watchpoint 3) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 4	000 Reserved 001 Reserved 010 CPU instruction address compare 5 (watchpoint 8) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 5	000 Reserved 001 Reserved 010 CPU instruction address compare 6 (watchpoint 9) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 80. L1SEL0 register field descriptions(Continued)

Field	Description
MUX 6	000 Reserved 001 Reserved 010 CPU instruction address compare 7 (watchpoint 14) 011 Reserved 100 Reserved 101 Reserved 110 CPU debug event input 2 (watchpoint 11) 111 CPU debug event output 3 (watchpoint 21)
MUX 7	000 Reserved 001 Reserved 010 CPU instruction address compare 8 (watchpoint 15) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 81 contains the L1SEL1 register field descriptions.

Table 81. L1SEL1 register field descriptions

Field	Description
MUX 8	000 Reserved 001 Reserved 010 CPU instruction address compare 1 (watchpoint 0) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 9	000 Reserved 001 Reserved 010 CPU instruction address compare 2 (watchpoint 1) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 10	000 Reserved 001 Reserved 010 CPU instruction address compare 3 (watchpoint 2) 011 Reserved 100 Reserved 101 Reserved 110 CPU debug event output 0 (watchpoint 12) 111 Reserved

Table 81. L1SEL1 register field descriptions(Continued)

Field	Description
MUX 11	000 Reserved 001 Reserved 010 CPU instruction address compare 4 (watchpoint 3) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 12	000 Reserved 001 Reserved 010 CPU instruction address compare 5 (watchpoint 8) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 13	000 Reserved 001 Reserved 010 CPU instruction address compare 6 (watchpoint 9) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 14	000 Reserved 001 Reserved 010 CPU instruction address compare 7 (watchpoint 14) 011 Reserved 100 Reserved 101 Reserved 110 CPU debug event output 1 (watchpoint 13) 111 Reserved
MUX 15	000 Reserved 001 Reserved 010 CPU instruction address compare 8 (watchpoint 15) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 82 contains the L1SEL2 register field descriptions.

Table 82. L1SEL2 register field descriptions

Field	Description
MUX16	000 Reserved 001 Reserved 010 CPU data address compare 1 (watchpoint 4) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 17	000 Reserved 001 Reserved 010 CPU data address compare 2 (watchpoint 5) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 18	000 Reserved 001 Reserved 010 CPU data address compare 3 (watchpoint 6) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 19	000 Reserved 001 Reserved 010 CPU data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 20	000 Reserved 001 Reserved 010 CPU data address compare 1 (watchpoint 4) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 21	000 Reserved 001 Reserved 010 CPU data address compare 2 (watchpoint 5) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 82. L1SEL2 register field descriptions(Continued)

Field	Description
MUX 22	000 Reserved 001 Reserved 010 CPU data address compare 3 (watchpoint 6) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 23	000 Reserved 001 Reserved 010 CPU data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 83 contains the L1SEL3 register field descriptions.

Table 83. L1SEL3 register field descriptions

Field	Description
MUX 24	000 Reserved 001 Reserved 010 CPU data address compare 1 (watchpoint 4) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 25	000 Reserved 001 Reserved 010 CPU data address compare 2 (watchpoint 5) 011 Reserved 100 Reserved 101 Reserved 110 CPU write to HID0 111 Reserved
MUX 26	000 Reserved 001 Reserved 010 CPU data address compare 3 (watchpoint 6) 011 Reserved 100 Reserved 101 Reserved 110 CPU write to HID1 111 Reserved

Table 83. L1SEL3 register field descriptions(Continued)

Field	Description
MUX 27	000 Reserved 001 Reserved 010 CPU data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 Reserved 110 CPU write to BUCSR 111 Reserved
MUX 28	000 Reserved 001 Reserved 010 CPU data address compare 1 (watchpoint 4) 011 Reserved 100 Reserved 101 Reserved 110 CPU write to SPEFSCR 111 Reserved
MUX 29	000 Reserved 001 Reserved 010 CPU data address compare 2 (watchpoint 5) 011 Reserved 100 Reserved 101 Reserved 110 CPU write to IVPR 111 Reserved
MUX 30	000 Reserved 001 Reserved 010 CPU data address compare 3 (watchpoint 6) 011 Reserved 100 Reserved 101 Reserved 110 CPU write to MSR 111 Reserved
MUX 31	000 Reserved 001 Reserved 010 CPU data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 84 contains the L1SEL4 register field descriptions.

Table 84. L1SEL4 register field descriptions

Field	Description
MUX 32	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Production device NAR full 111 GTM watchpoint 0
MUX 33	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Production device NAR partial full 111 GTM watchpoint 1
MUX 34	000 Reserved 001 Reserved 010 CPU memory protection unit (watchpoint 27) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 2
MUX 35	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 3
MUX 36	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 4
MUX 37	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 5

Table 84. L1SEL4 register field descriptions(Continued)

Field	Description
MUX 38	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 6
MUX 39	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 7

Table 85 contains the L1SEL5 register field descriptions.

Table 85. L1SEL5 register field descriptions

Field	Description
MUX 40	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 0 111 GTM watchpoint 8
MUX 41	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 1 111 GTM watchpoint 9
MUX 42	000 CPU interrupt (watchpoint 16) 001 CPU critical interrupt (watchpoint 18) 010 CPU data trace control range 1 (watchpoint 29) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 2 111 GTM watchpoint 10

Table 85. L1SEL5 register field descriptions(Continued)

Field	Description
MUX43	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 3 111 GTM watchpoint 11
MUX 44	000 Reserved 001 Reserved 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
MUX 46	000 CPU return (watchpoint 17) 001 CPU critical return (watchpoint 19) 010 CPU data trace control range 2 (watchpoint 30) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 6 111 Reserved
MUX 47	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 7 111 Reserved

Table 86 contains the L1SEL6 register field descriptions.

Table 86. L1SEL6 register field descriptions

Field	Description
MUX 48	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 8 111 Reserved
MUX 49	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 9 111 Reserved
MUX 50	000 CPU context load (using e_lmvsprw instruction) 001 CPU context restore (using e_stmvsprw instruction) 010 CPU data trace control range 3 (watchpoint 31) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 10 111 Reserved
MUX 51	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 11 111 Reserved
MUX 52	000 Reserved 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

Table 86. L1SEL6 register field descriptions(Continued)

Field	Description
MUX 54	000 CPU interrupt current priority match to C2PIS 001 CPU write to ME bits of MSR 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

Table 87 contains the L1SEL7 register field descriptions.

Table 87. L1SEL7 register field descriptions

Field	Description
MUX 56	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTI0
MUX 57	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTI1
MUX 58	000 CPU write to EE bits of MSR 001 CPU write to CE bits of MSR 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTO0

Table 87. L1SEL7 register field descriptions(Continued)

Field	Description
MUX 59	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTO1
MUX 60	000 Reserved 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
MUX 62	000 CPU processor exception vector match to C2PEVP 001 CPU write to DE bits of MSR 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

11.3 Calibration interface

The device powers up with the JTAG module having control of the pins. The interface is supported entirely in hardware for JTAG mode, with no software or system overhead required to support interface traffic.

Table 88. Reference links to related information

Related module	Reference
JTAG Master (JTAGM)	*Chapter 62: JTAG Master (JTAGM)*
Development Tool Semaphore (DTS)	*Chapter 64: Development Trigger Semaphore (DTS)*

11.3.1 JTAG Master (JTAGM)

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 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 module is connected in the device to allow these five signals to connect to the JTAGC as if the JTAG data was coming from an external tool. JTAGM generates all required JTAG scan chains to allow software and high speed serial communication access to all JTAG mapped resources.

11.3.2 Development Trigger Semaphore (DTS)

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways this module can be used, including as a component of an external real-time data acquisition system.

Note: When used as a component of a triggered data acquisition system, Nexus read/write access is via the JTAG interface of the Nexus debug port and is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2011 Nexus standards, which use the Nexus Auxiliary port.

11.4 Debug over CAN

As well as supporting debug via the JTAG interface, the device supports the use of a CAN interface for debug. This allows debug of application hardware where access to the dedicated JTAG interface signals is not easily available. The debug over CAN mechanism is intended to provide basic debug functionality with some reduction in bandwidth and features compared to use of the dedicated JTAG interface.

Use of the CAN interface for debug purposes requires the use of some M_CAN, eDMA, and JTAGM resources. Some initialization of these resources via software is required, but once this initialization has completed, debug over CAN is possible without any further software overhead.

As the debug over CAN scheme generates internal JTAG messages based on received CAN data, all JTAG clients and included debug resources are accessible. Basic device trace capability is also possible by configuring the device trace hardware to stream to a device overlay/trace RAM, which can be read later using debug over CAN.

The debug over CAN scheme supports the following features:

- Supports operation through M_CAN0 or M_CAN1 interfaces
  - Makes use of M_CAN debug enhancements
  - Debug message selected by filter configuration EFEC[2:0]=111
- Allows debug of hardware where JTAG access is not available
- Debug traffic carried over application CAN bus
- Debug traffic on CAN coexists with application traffic
- No software overhead after initialization
- flexible selection of CAN identifiers for debug use
- Access to all JTAG debug facilities, includes CPU run control

11.5 Nexus Aurora

The Aurora information can be used to reconstruct events or operations that occurred inside the microcontroller. Each Nexus message is tagged with the source client identifier and the type of message. The actual trace information that is available depends on the Nexus client and can usually be enabled individually (both the type of messages and which clients). *Table 89* shows a few examples of Nexus messages.

Table 89. Types of nexus messages

Message type	Description
Program Trace	Nexus Program trace messages transmit any discontinuities in the program flow, such as branches and interrupts. Multiple types of messages can be generated.
Ownership Trace	Ownership Trace provides information on process identification changes.
Watchpoint Trace	Watchpoint messages are generated anytime a watchpoint match occurs in the program or data flow, or as a result of SPU actions. These can be programmed for many types of events within the MCU.
Device Identification	The device identification message allows information about the MCU to be transmitted on power up to allow tools to identify the target system MCU type.
Debug Status	Debug status message provides additional information that may be needed to reconstruct software execution. Information such as if the device has entered debug mode or low power mode.
Data Acquisition	Data Acquisition messages are an optional feature that allows software control of information to be transmitted.
Error	Error messages are transmitted when an error condition occurs, such as internal buffer overflows.

The Nexus trace information is output to internal memory.

11.5.1 Nexus Aurora register interface overview

The Nexus blocks do not have memory-mapped registers. The Nexus registers are accessed by the development tool via the JTAG port using a two step process. First, the specific block is selected by loading the corresponding ACCESS_AUX instruction as described in the JTAGC chapter. After the block is selected, the Nexus registers are enabled by loading the enable instruction to the JTAGC Instruction Register (IR). The enable

instructions for the different Nexus clients are shown in *Table 90*. After the enable instruction is received, the development tool has access to the Nexus registers of the selected client.

Table 90. Nexus client JTAG enable instructions

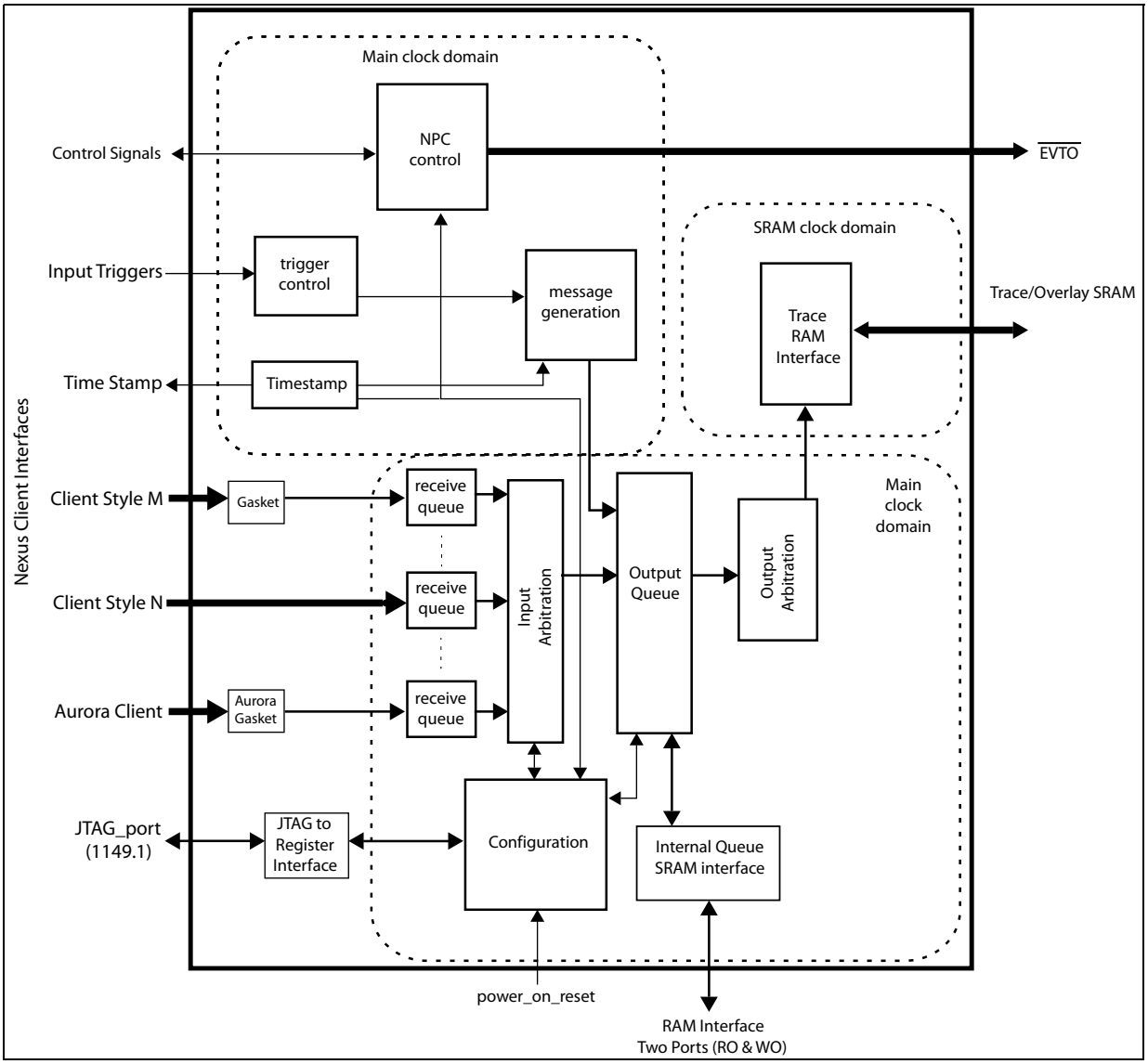
Module	Enable instruction	Opcode
Core (e200z2)	CORE_ENABLE	0x7C (10 bits)
GTM	NEXUS_ENABLE	0x0 (4 bits)
NAR		
SPU		

11.5.2 Nexus Aurora Router (NAR)

All Nexus trace information from each of the Nexus clients is managed by the Nexus Aurora Router (NAR). The NAR takes the parallel trace information from each of the connected clients, buffers it to avoid overruns, and routes the information to the internal overlay RAM.

8 KB of trace SRAM is available in the flash module. A 2 KB NAR output queue is located on the device. *Figure 43* shows the NAR block diagram.

Figure 43. NAR block diagram



The PD NAR module has two clients with client index 1and 3. All other client indexes are not connected. *Table 91* contains the client connections and the corresponding NAR client index. The NAR client index is used in the NAR_STCR, NAR_CSSR, and NAR_CDR registers. The source trace ID is used in the NAR_SFR.

Table 91. PD NAR clients

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
1	GTM	0b0111 (0x7)	3
2	—	—	—
3	Core	0b0010 (0x2)	1
4	—	—	—
5	—	—	—

Table 91. PD NAR clients(Continued)

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
6	—	—	—
7	—	—	—
8	—	—	—
9	—	—	—
10	—	—	—
11	—	—	—
12	—	—	—
13	—	—	—
14	—	—	—
15	—	—	—
16	—	—	—

The NAR is limited writing trace information to the Overlay RAM. The PD NAR only has access to the 8 KB internal Overlay RAM. Therefore the PD NAR trace memory bus base address registers (NAR_TBALO and NAR_TBAHI) must specify a base address in the range 0x0D000000–0x0D001FFF.

11.6 Nexus clients

Nexus supports two trace clients within the device for transmitting information through JTAG and CAN via the internal overlay Ram.

Table 92. Reference links to related information

Related module	Reference
e200z2 Nexus Support	*Chapter 66: e200z215An3 Nexus 3 Module*
GTM Development Interface (GTMDI)	*Chapter 42: GTM Development Interface (GTMDI)*

11.6.1 e200z2 Nexus 3

The e200z215An3 Nexus 3 modules provide real-time development capabilities for the e200z2 processor in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

12 Core e200z215An3 description

12.1 Overview of the e200z215An3 core

The e200z215An3 is a single-issue 32-bit *PowerISA 2.06* VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z215An3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in PowerISA 2.06, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

The EFPU2 APU supports 32-bit IEEE-754 single-precision floating-point formats and operates in a pipelined fashion. The 32-bit general-purpose register file is used for source and destination operands. There is a unified storage model for scalar single-precision floating-point data types of 32 bits and the normal integer type. Low latency floating-point add, subtract, mixed add/subtract, sum, diff, min, max, multiply, multiply-add, multiply-sub, divide, square root, compare, and conversion operations are provided. Most operations can be pipelined.

A Lightweight Signal Processing Extension (LSP) APU is provided to support real-time SIMD fixed-point embedded numerics operations using the general-purpose registers. All arithmetic instructions that execute in the core operate on data in the general purpose registers (GPRs). The GPRs support register pairing in order to support vector instructions defined by the LSP APU. These instructions operate on a vector pair of 16-bit or 32-bit data types, and deliver vector and scalar results. A wide variety of data manipulation, multiply, multiply/accumulate, and dot product instructions along with specialized memory access instructions allow a sustained throughput of up to two one 16 × 16-bit multiply/accumulate operations per cycle.

The e200z215An3 processor integrates an integer execution unit, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining three read and two write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

e200z215An3 contains a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

12.2 EFPU2 Floating-Point Unit features

The EFPU2 embedded floating-point unit supports pipelined operation of most floating-point operations, allowing a throughput of one FP operation per cycle, and supports a variety of operations:

- Supports IEEE754 single-precision data format
- Supports IEEE754 half-precision format for conversion <-> single-precision format
- EFPU2 instructions utilize the 32-bit GPRs for minimized context save/restore overhead
- Provides low latency pipelined floating-point add, subtract, add/sub, sub/add, multiply, multiply-add, min, max, absolute value, compare, and conversion instructions with single-cycle throughput
- Default results mode for real-time applications allows for exception-free operations when underflow or overflows occur
- Trap Enable controls allow for trapping of various operation exceptions and emulation of IEEE754 boundary case results

Floating-point Divide and Square root logic operating in 13 (FP div) and 11 (FP sqrt) clock cycles

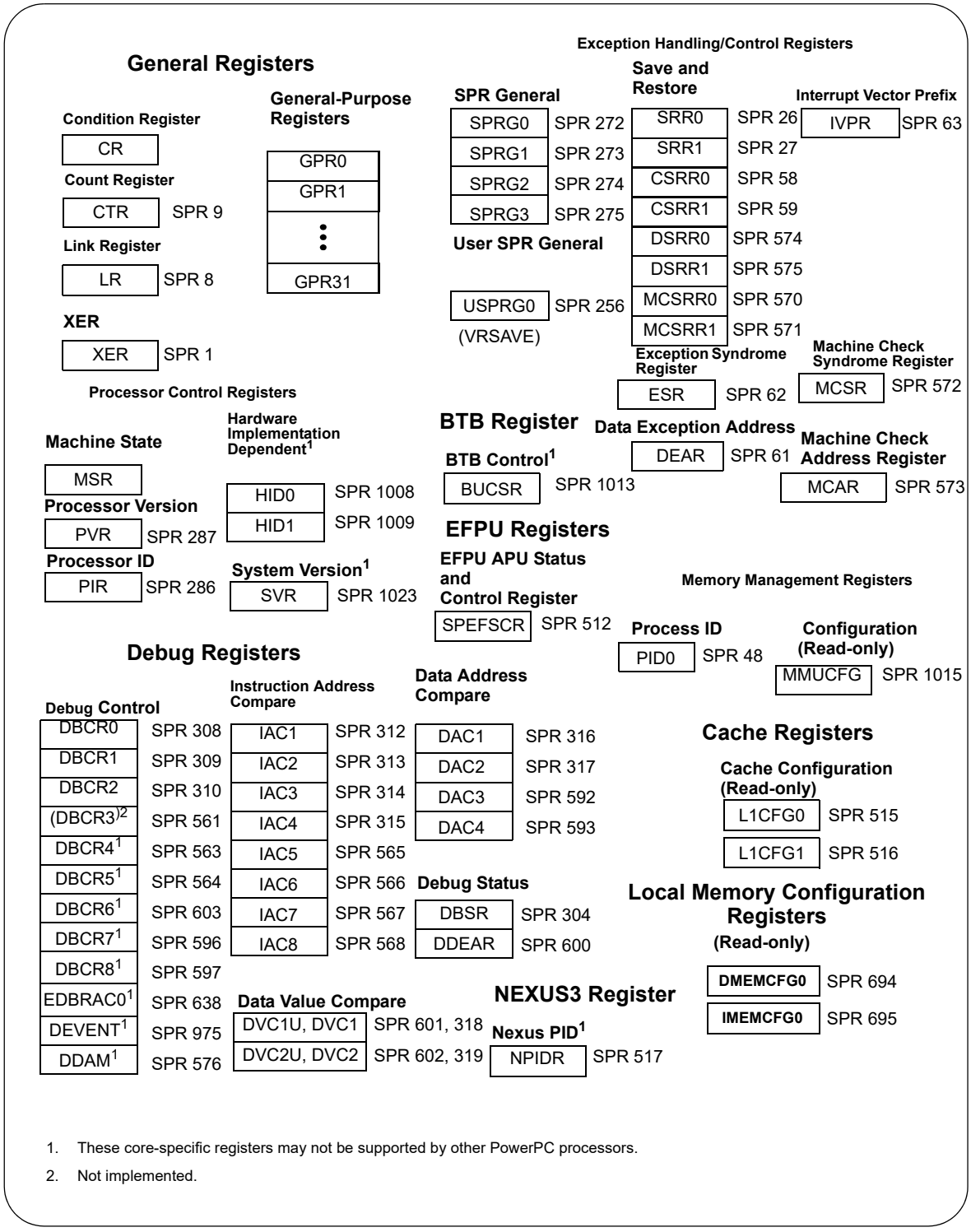
12.3 Register model

This section describes the registers implemented in the e200z215An3 core. The PowerISA 2.06 architecture defines register-to-register operations for all computational instructions.

e200z215An3 extends usage of the general purpose registers to support EFPU operations on the 32-bit GPR registers.

Figure 44 and *Figure 45* show the complete e200z215An3 register set, all of which are accessible in supervisor mode. *Figure 46* and *Figure 47* show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

Figure 44. e200z215An3 Supervisor Mode Programmer's Model SPRs



1. These core-specific registers may not be supported by other PowerPC processors.
2. Not implemented.

Figure 45. e200z215An3 Supervisor Mode Programmer's Model DCRs and PMRs

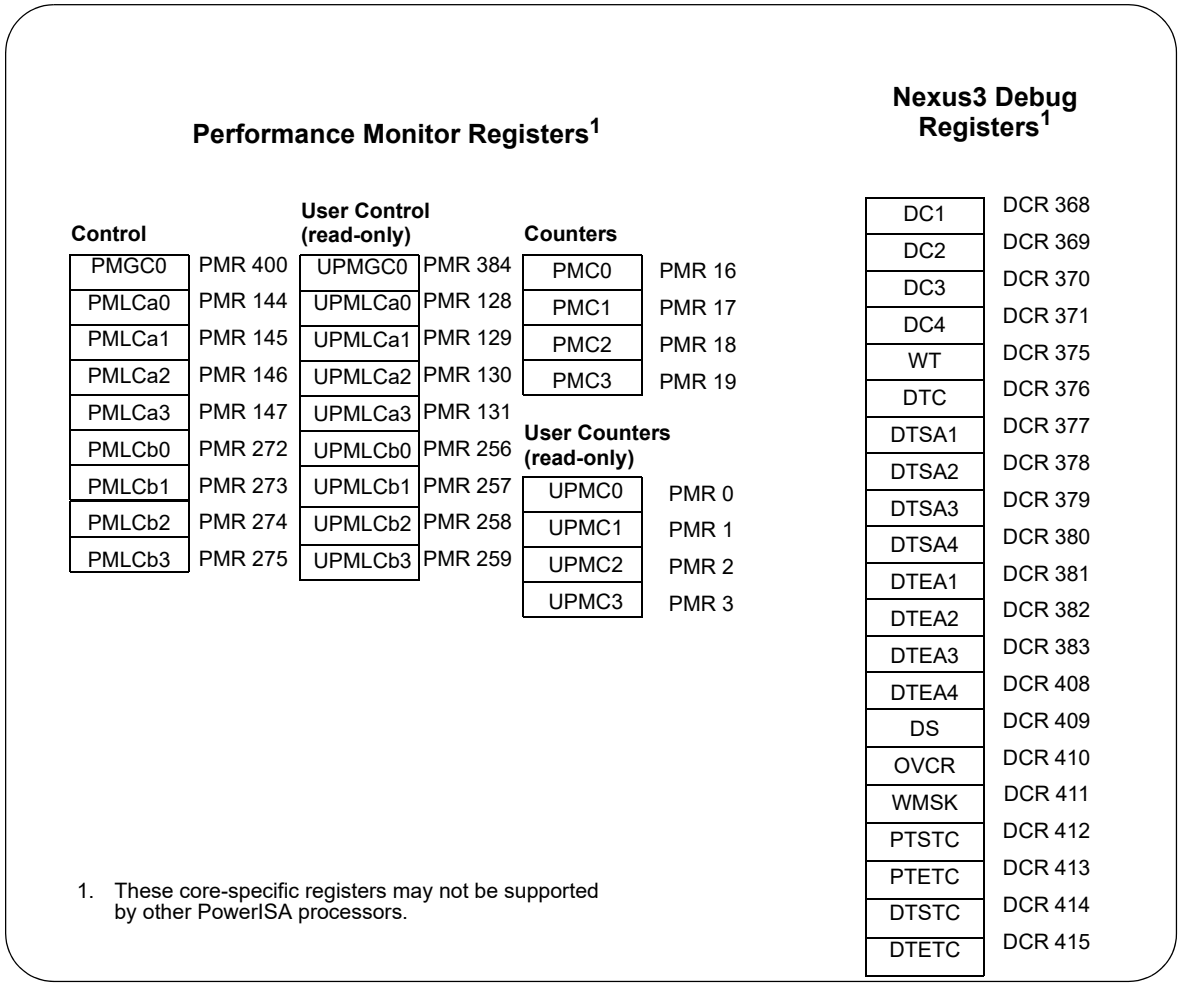


Figure 46. e200z215An3 User Mode Programmer's Model SPRs

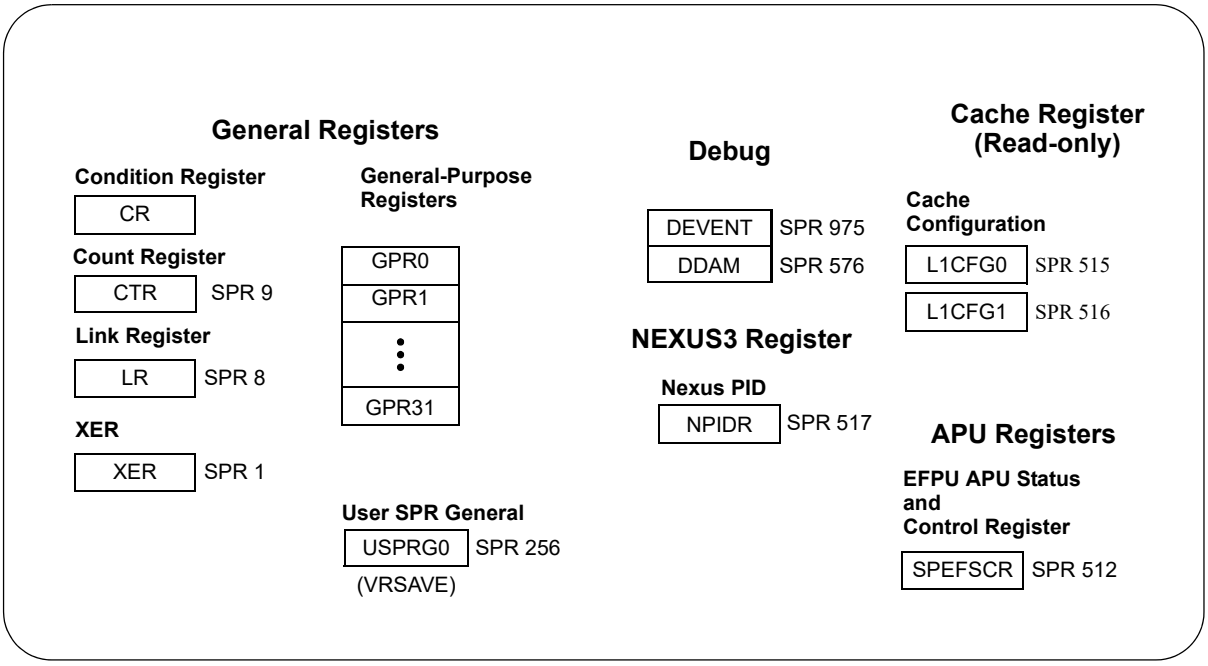
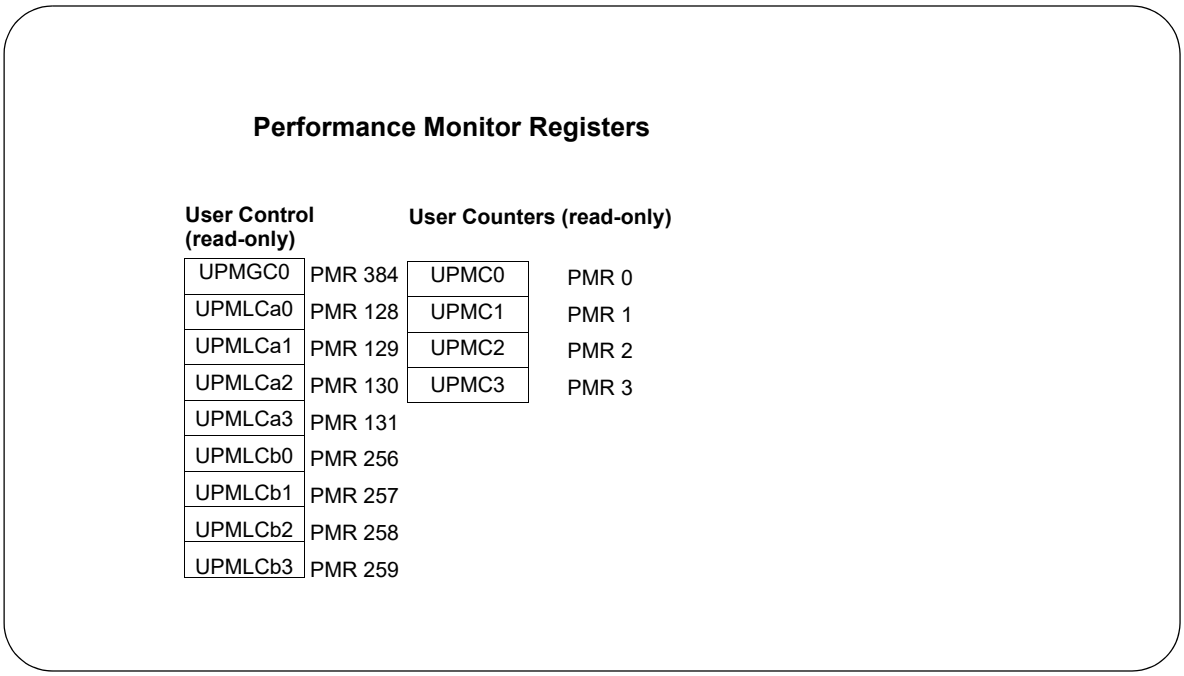


Figure 47. e200z215An3 User Mode Programmer's Model PMRs



12.4 Single-issue operation

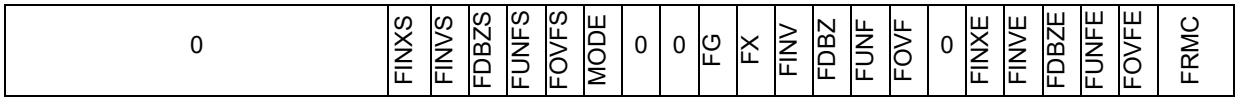
The instruction issue unit attempts to issue an instruction to the execution units each cycle. Source operands for each of the instructions are provided from the GPRs or from the operand feed-forward muxes. Data or resource hazards may create stall conditions that cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. Up to two results can be simultaneously written.

12.5 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point operations use the SPEFSCR. The SPEFSCR is implemented as special-purpose register (SPR) number 512 and is read and written by the mfspr and mtspr instructions. The SPEFSCR is shown in *Figure 48*.

Figure 48. EFPU Status and Control Register (SPEFSCR)



SPR 512; Read/Write; Reset: 0x0

The SPEFSCR bits are defined in *Table 93*.

Table 93. EFPU Status and Control Register field descriptions

Bits	Name	Description
0:9 (32:41)	—	Reserved
10 (42)	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF = 1), but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1), but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
11 (43)	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
12 (44)	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
13 (45)	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.

Table 93. EFPU Status and Control Register field descriptions(Continued)

Bits	Name	Description
14 (46)	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
15 (47)	MODE	Embedded Floating-point Operating Mode This bit controls the operating mode of the EFPU. e200z215An3 supports only mode 0. Software should read the value of this bit after writing it to determine if the implementation supports the selected mode. Implementations will return the value written if the selected mode is a supported mode, otherwise the value read will indicate the hardware supported mode. 0 Default hardware results operating mode 1 IEEE754 hardware results operating mode (not supported by Zen)
16 (48)	—	Reserved
17 (49)	—	Reserved
18 (50)	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19 (51)	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler.FX is zeroed if a Floating-point Data Exception occurs.
20 (52)	FINV	Embedded Floating-point Invalid Operation / Input error In mode 0, the FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0. In mode 1, the FINV bit is set on an IEEE754 invalid operation (IEEE754-1985 sec7.1).
21 (53)	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite nonzero number.
22 (54)	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23 (55)	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24 (56)	—	Reserved

Table 93. EFPU Status and Control Register field descriptions(Continued)

Bits	Name	Description
25 (57)	FINXE	Embedded Floating-point Inexact Exception Enable If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG | FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF = 1) but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1 or FUNFH = 1) but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. 0 Exception disabled 1 Exception enabled
26 (58)	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
27 (59)	FDBZE	Embedded Floating-point Divide by Zero Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
28 (60)	FUNFE	Embedded Floating-point Underflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
29 (61)	FOVFE	Embedded Floating-point Overflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
30:31 (62:63)	FRMC	Embedded Floating-point Rounding Mode Control 00 Round to Nearest 01 Round toward Zero 10 Round toward +Infinity 11 Round toward –Infinity

12.6 Exceptions

Interrupts implemented in e200z215An3 and the exception conditions that cause them are listed in *Table 94*.

Table 94. Exceptions and conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	Reset
Critical Input	0x00(1)	**p_critint_b** is asserted and MSR_{CE} = 1.
Machine check	0x10	1. **p_mcp_b** transitions from negated to asserted 2. ISI or Bus Error on first instruction fetch for an exception handler 3. External bus error 4. Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40(2)	Interrupt Controller interrupt and MSR_{EE} = 1
Alignment	0x50	1. **lmw**, **stmw** not word aligned 2. **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned **3. dcbz**
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 1
EFPU Data Exception	0xA0	See section "Embedded Floating-point Data Exception" in the Power Architecture core reference manual relative to this core.
EFPU Round Exception	0xB0	See section "Embedded Floating-point Round Exception" in the Power Architecture core reference manual relative to this core.
TBD	0xC0–0xF0	Reserved for future processor use

1. Autovectored Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

12.6.1 Exception Syndrome Register (ESR)

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.

0	PIL	PPR	PTR	FP	ST	0	0	0	BO	0	0	SPV	0	VLEMI	0	0	0
0:3	4	5	6	7	8	9:11	12	13	14	15	16:23	24	25	26	27:29	30	31

SPR 62; Read/Write; Reset: 0x0

Figure 49. Exception Syndrome Register (ESR)

The ESR bits are defined in *Table 95*.

Table 95. ESR field descriptions

Bit(s)	Name	Description	Associated interrupt type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception For core e200z215An3, PIL is used for both illegal and unimplemented instructions.	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	—	Reserved	—
13	—	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	—	Reserved	—
16:23	—	Reserved	—
24	SPV	EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—

Table 95. ESR field descriptions(Continued)

Bit(s)	Name	Description	Associated interrupt type
30	—	Reserved	—
31	—	Reserved	—

12.6.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z215An3 MSR is shown in *Figure 50*.

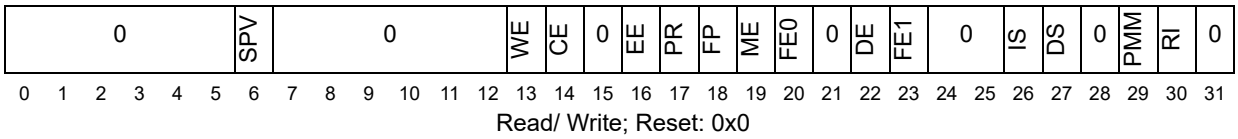


Figure 50. Machine State Register (MSR)

The MSR bits are defined in *Table 96*.

Table 96. MSR field descriptions

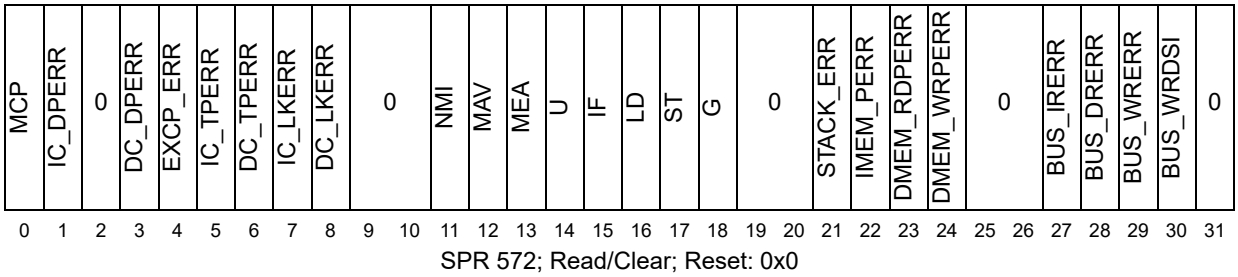
Bit(s)	Name	Description
0:5	—	Reserved
6	SPV	SP/Embedded FP/Vector available 0 SP/Embedded FP Double/The processor cannot execute SP/Embedded FP Double or Vector Unit instructions. 1 SP/Embedded FP Double/The processor can execute SP/Embedded FP Double or Vector Unit instructions. **Note:** For Zen z215An3, these units are not supported in hardware, and an illegal instruction exception will be generated for attempted execution of these instructions regardless of the setting of SPV. SPV is ignored, but cleared on exceptions.
7:12	—	Reserved
13	WE	Wait State (Power management) enable **Note:** This bit is implementation dependent and is being phased out. It is to be removed in future release. Currently, it is implemented as a writeable bit, and is ignored, but cleared on exceptions.
14	CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.
15	—	Reserved
16	EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.
17	PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.

Table 96. MSR field descriptions(Continued)

Bit(s)	Name	Description
18	FP	Floating-Point Available 0 Floating point unit is unavailable. The processor cannot execute floating-point instructions, including floating-point loads, stores, and moves. 1 Floating Point unit is available. The processor can execute floating-point instructions. **Note:** For Zen z215An3, the PowerISA 2.06 floating point unit is not supported in hardware, and an Illegal Instruction exception will be generated for attempted execution of PowerISA 2.06 floating point instructions regardless of the setting of FP. FP is ignored, but cleared on exceptions.
19	ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
20	FE0	Floating-point exception mode 0 (not used by Zen) **Note:** For Zen z215An3, the PowerISA 2.06 floating point unit is not supported in hardware, thus FE0 is ignored, but cleared on exceptions.
21	—	Reserved
22	DE	Debug Interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23	FE1	Floating-point exception mode 1 (not used by Zen) **Note:** or Zen z215An3, the PowerISA 2.06 floating point unit is not supported in hardware, thus FE1 is ignored, but cleared on exceptions.
24:25	—	Reserved
26	IS	Instruction Address Space 0 The processor directs all instruction fetches to address space 0. 1 The processor directs all instruction fetches to address space 1. **Note:** For Zen z215An3, Address Spaces are not supported, thus the IS bit is ignored, but cleared on exceptions.
27	DS	Data Address Space 0 The processor directs all data storage accesses to address space 0. 1 The processor directs all data storage accesses to address space 1. **Note:** For Zen z215An3, Address Spaces are not supported, thus the DS bit is ignored, but cleared on exceptions.
28	—	Reserved
29	PMM	PMM Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCa*n* Performance Monitor registers, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken.
31	—	Reserved

12.6.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome Register (MCSR) to differentiate between machine check conditions. The MCSR is shown in *Figure 51*.



SPR 572; Read/Clear; Reset: 0x0

Figure 51. Machine Check Syndrome Register (MCSR)

Table 97 describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as “write 1 to clear.” Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once MSR_{ME} = 1.

Table 97. Machine Check Syndrome Register (MCSR) field descriptions

Bit	Name	Description	Exception type(1)	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR(2)	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved	—	—
3	DC_DPERR(3)	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR(2)	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR(3)	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR(2)	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—

Table 97. Machine Check Syndrome Register (MCSR) field descriptions(Continued)

Bit	Name	Description	Exception type(1)	Recoverable
8	DC_LKERR(3)	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
9:10	—	Reserved	—	—
11	NMI	NMI input pin	NMI	—
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV = 1,MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address(4)	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to a stack limit check error, or an external bus error.	Error Report	Precise
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to a stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus.	Error Report	Precise
19:20	—	Reserved	—	—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR(5)	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise

Table 97. Machine Check Syndrome Register (MCSR) field descriptions(Continued)

Bit	Name	Description	Exception type(1)	Recoverable
23	DMEM_RDPERR(6)	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR(6)	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved	—	—
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved	—	—

1. The Exception Type indicates the exception type associated with a given syndrome bit.

   — “Error Report” indicates that this bit is only set for error report exceptions that cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

   — “Status” indicates that this bit is provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

   — “NMI” indicates that this bit is only set for the non-maskable interrupt type exception, which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a ‘1’ to the bit position.

   — “Async Mchk” indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any “Async Mchk” bit is set in the MCSR, a machine check interrupt will occur if $MSR_{ME} = 1$. If $MSR_{ME} = 0$, the machine check exception will remain pending. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

2. Will not be set by e200z215An3 since no instruction cache is present.
3. Will not be set by e200z215An3 since no data cache is present.
4. Note that since no address translation mechanism is present on e200z215An3, the MEA value is always set to ‘0’.
5. Will not be set by e200z215An3 since no local instruction memory is present.
6. Will not be set by e200z215An3 since no local data memory is present.

12.6.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in *Figure 52*.

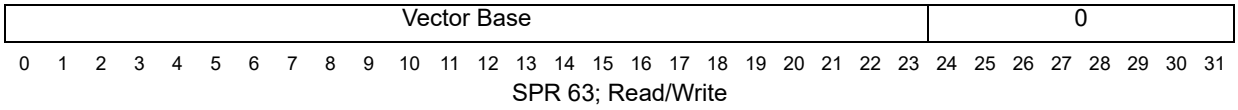


Figure 52. Zen Interrupt Vector Prefix Register (IVPR)

The IVPR fields are defined in *Table 98*.

Table 98. IVPR field descriptions

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256-byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

12.6.5 Interrupt Definitions

12.6.5.1 Critical Input Interrupt (offset 0x00)

A Critical Input exception is signaled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

Table 99 lists register settings when a Critical Input interrupt is taken.

Table 99. Critical Input Interrupt—register settings

Register	Setting description		
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
CSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME — FE0 0 DE —/0(1)	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x00 (autovectored) $IVPR_{0:15}$ || ($IVPR_{16:29}$ | **p_voffset[0:13]**) || 2'b00 (non-autovectored)		

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but it can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

12.6.5.2 Machine Check Interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in *Table 100*.

Table 100. Machine Check Interrupt—register settings

Register	Setting description		
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.		
MCSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME 0 FE0 0 DE 0/—(1)	FE1 0 IS 0 DS 0 PMM 0 RI 0
ESR	Unchanged		
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.		
Vector	$IVPR_{0:23}$ || 0x10		

1. Clearing of DE is optionally supported by control in HID0.

12.6.5.3 Data Storage Interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

Table 101 lists register settings when a DSI is taken.

Table 101. Data Storage Interrupt—register settings

Register	Setting description
SRR0	Set to the effective address of the excepting load/store instruction.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table 101. Data Storage Interrupt—register settings(Continued)

Register	Setting description		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Access:	[ST], [SPV], VLEMI. All other bits cleared.	
MCSR	Unchanged		
DEAR	For Access Control exceptions, set to the effective address of the access that caused the violation.		
Vector	$IVPR_{0:23}$ || 0x20		

12.6.5.4 Instruction Storage Interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

Table 102 lists register settings when an ISI is taken.

Table 102. Instruction Storage Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x30		

12.6.5.5 External Input Interrupt (offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

Table 103 lists register settings when an External Input interrupt is taken.

Table 103. External Input Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x40 (autovectored) IVPR0:15 || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)		

12.6.5.6 Alignment Interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.

Table 104 lists register settings when an alignment interrupt is taken.

Table 104. Alignment Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	[ST], VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Set to the effective address of a byte of the load or store access causing the violation.		
Vector	$IVPR_{0:23}$ || 0x50		

12.6.5.7 Program Interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

The e200z215An3 will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

The e200z215An3 will invoke a Privileged Instruction program exception on attempted execution of the following instructions when MSR_{PR} = 1 (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions that specify a SPRN value with $SPRN_5$ = 1 (even if the SPR is undefined)

The e200z215An3 will invoke an Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any PowerISA 2.06 floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause an illegal instruction program exception.

Table 105 lists register settings when a Program interrupt is taken.

Table 105. Program Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Illegal, Unimplemented: Privileged: Trap:	PIL, VLEMI. All other bits cleared. PPR, VLEMI. All other bits cleared. PTR, VLEMI. All other bits cleared.	
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x60		

12.6.5.8 System Call Interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

Table 106 lists register settings when a System Call interrupt is taken.

Table 106. System Call Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the instruction *following* the system call instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x80		

12.6.5.9 Debug Interrupt (offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM}$ = 1 (internal debug mode) and MSR_{DE} = 1).

Table 107 lists register settings when a Debug interrupt is taken.

Table 107. Debug Interrupt—register settings

Register	Setting description		
DSRR0(1)	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP. Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
DSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE —/0(2) EE —/0(2) PR 0	FP 0 ME — FE0 0 DE 0	FE1 0 IS 0 DS 0 PMM 0 RI —

Table 107. Debug Interrupt—register settings(Continued)

Register	Setting description	
DBSR[3]	Unconditional Debug Event: Instr. Complete Debug Event: Branch Taken Debug Event: Interrupt Taken Debug Event: Critical Interrupt Taken Debug Event: Trap Instruction Debug Event: Instruction Address Compare: Data Address Compare: Debug Notify Interrupt: Return Debug Event: Critical Return Debug Event: External Debug Event: Performance Monitor Debug Event: and optionally, an Imprecise Debug Event flag	UDE ICMP BRT IRPT CIRPT TRAP {IAC} {DACR, DACW} DNI RET CRET {DEVT1, DEVT2} PMI {IDE}
ESR	Unchanged	
MCSR	Unchanged	
DEAR	Unchanged	
Vector	$IVPR_{0:23}$ || 0x90	

1. Assumes that the Debug interrupt is precise
2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

12.6.5.10 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

Table 108 lists register settings when a System Reset interrupt is taken.

Table 108. System Reset Interrupt—register settings

Register	Setting description		
CSRR0	Undefined		
CSRR1	Undefined		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME 0 FE0 0 DE 0	FE1 0 IS 0 DS 0 PMM 0 RI 0

Table 108. System Reset Interrupt—register settings(Continued)

Register	Setting description
ESR	Cleared
DEAR	Undefined
Vector	[**p_rstbase[0:29]**] || 2'b00

12.6.5.11 Embedded Floating-point Data Interrupt (offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

Table 109 lists register settings when an EFPU Floating-point Data interrupt is taken.

Table 109. Embedded Floating-point Data Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the excepting EFPU instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	SPV, VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0xA0		

12.6.5.12 Embedded Floating-point Round Interrupt (offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

Table 110 lists register settings when an EFPU Floating-point Round interrupt is taken.

Table 110. Embedded Floating-point Round Interrupt—register settings

Register	Setting description		
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME — FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI —

Table 110. Embedded Floating-point Round Interrupt—register settings(Continued)

Register	Setting description
ESR	SPV, VLEMI. All other bits cleared.
MCSR	Unchanged
DEAR	Unchanged
Vector	$IVPR_{0:23}$ || 0xB0

12.6.5.13 Performance Monitor Interrupt (offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- $PMLCax_{CE}$ = 11; that is, for the given counter the overflow condition is enabled.
- $PMCx_{OV}$ = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

Table 111 lists register settings when an performance monitor interrupt is taken.

Table 111. Performance Monitor Interrupt—Register Settings

Register	Setting description		
SRR0/ DSRR0(1)	Set to the effective address of the next instruction to be executed.		
SRR1/ DSRR1(1)	Set to the contents of the MSR at the time of the interrupt.		
MSR	SPV 0 WE 0 CE —/0(2) EE 0/—(3) PR 0	FP 0 ME — FE0 0 DE —/0(4)	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x70		

1. DSRR0/1 are used if $PMGC0_{UDI}$ = 1
2. CE is cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLRCE}$ = 1
3. EE is not cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLREE}$ = 0
4. DE is cleared if $PMGC0_{UDI}$ = 1

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 seven ports with 16 bits of bidirectional, general-purpose input and output signals. It supports three external interrupts with trigger event configuration. *Figure 53* shows the block diagram of the SIUL2 module 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.

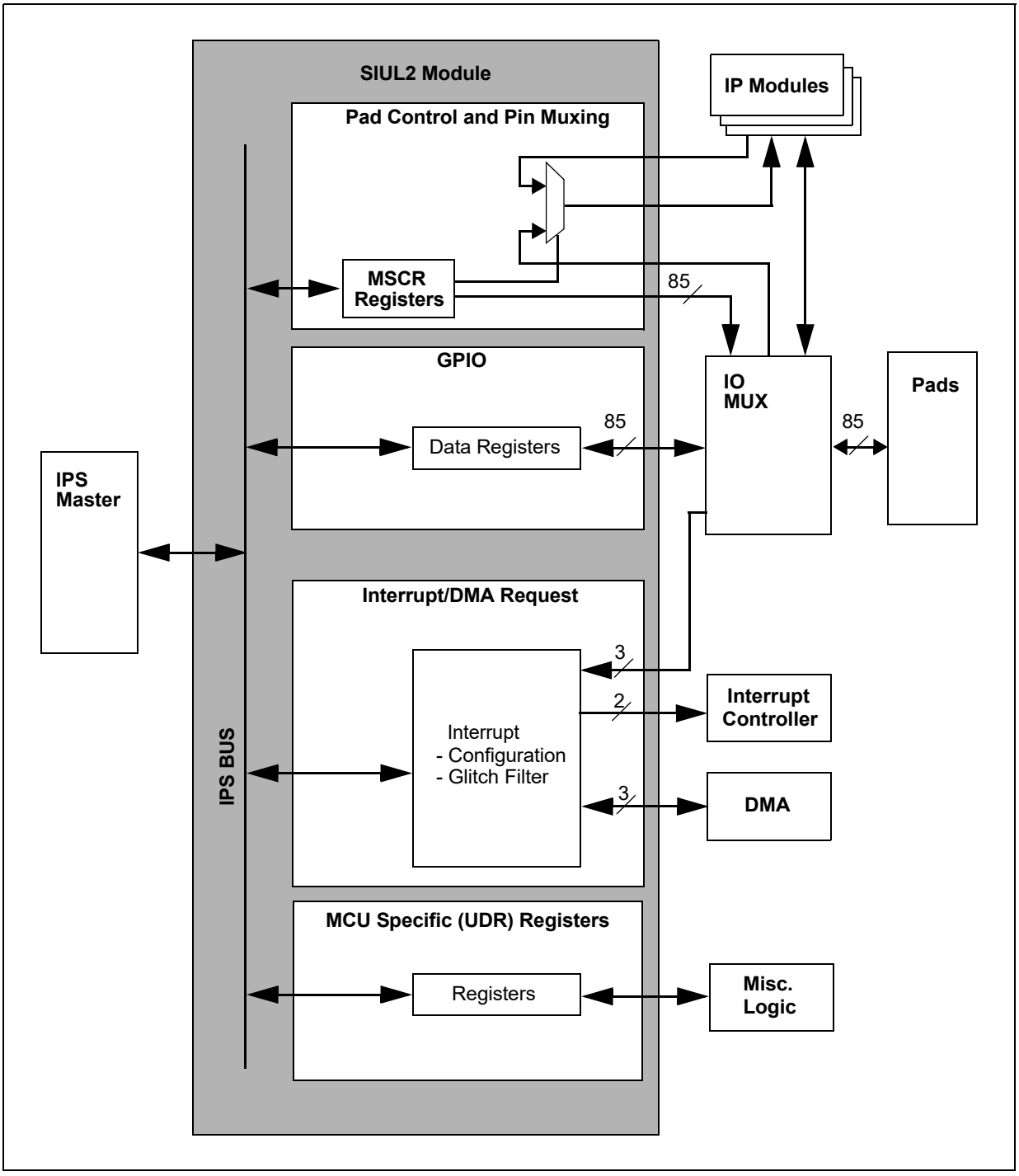
- When configured as output, you can write to an internal register to control the state driven on the associated output pad.
- When configured as input, you can detect the state of the associated pad by reading the value from an internal register.
- When configured as input and output, the pad value can be read back, which can be used as a method of checking if the written value appeared on the pad.

The different access mechanisms to the GPIO data registers aid in software development. These differing mechanisms allow support for port access or for bit manipulation without the need to use read-modify-write operations or a Power Architecture reservation access.

- 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

The external interrupt sources can be configured on MCU level to be used with any chip pad. The three 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 three EIRQ pins in the device packages.

Figure 53. SIUL2 block diagram



13.1.2 Features

The SIUL2 supports these distinctive features:

- 7 GPIO ports with data control
  - Drive data to up to 16 independent I/O channels
  - Sample data from up to 16 independent I/O channels
- Two 16-bit registers can be read/written with one access to a 32-bit port if needed.
- External interrupt/DMA request support with:
  - 2 system interrupt vectors for 3 interrupt sources connected to EIRQ pins with independent interrupt mask
  - 3 programmable digital glitch filters (one for each EIRQ pin)
  - 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 85 functional pads. 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. The following registers can be protected:

- SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)
- SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)
- SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)
- SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)
- SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)
- SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0–SIUL2_IFMCR31)
- SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)
- SIUL2 I/O pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511)
- SIUL2 Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR_MUX_512–SIUL2_MSCR_MUX_1023)

Details regarding the protected registers at specific addresses can be found in *Section 6.9.3.1: Register protection (REG_PROT) configuration* and *Chapter 67: Register Protection (REG_PROT)*.

13.2 Memory map and register description

This section provides a detailed description of all registers accessible in the SIUL2 module.

13.2.1 Memory map

Table 112 gives an overview of the SIUL2 registers implemented.

Table 112. SIUL2 memory map

Address offset	Register name	Access	Size (bits)	Supported access (bit)	Location
0x0000	Reserved				
0x0004	SIUL2 MCU ID Register #1 (SIUL2_MIDR1)	R	32	32	*on page 238*
0x0008	SIUL2 MCU ID Register #2 (SIUL2_MIDR2)	R	32	32	*on page 238*
0x000C–0x000F	Reserved				
0x0010	SIUL2 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0)	R/W	32	32/16/8	*on page 240*
0x0014–0x0017	Reserved				
0x0018	SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)	R/W	32	32/16/8	*on page 241*
0x001C–0x001F	Reserved				
0x0020	SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)	R/W	32	32/16/8	*on page 242*
0x0024	Reserved				
0x0028	SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)	R/W	32	32/16/8	*on page 243*
0x002C	Reserved				
0x0030	SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)	R/W	32	32/16/8	*on page 244*
0x0034	Reserved				
0x0038	SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)	R/W	32	32/16/8	*on page 244*
0x003C	Reserved				
0x0040–0x00BC	SIUL2 Interrupt Filter Maximum Counter Register 0–Interrupt Filter Maximum Counter Register 31 (SIUL2_IFMCR0–SIUL2_IFMCR31)	R/W	32	32	*on page 245*

Table 112. SIUL2 memory map(Continued)

Address offset	Register name	Access	Size (bits)	Supported access (bit)	Location
0x00C0	SIUL2 Interrupt Filter Clock Prescaler (SIUL2_IFCPR)	R/W	32	32	on page 246
0x00C4–0x023F	Reserved				
0x0240–0xA3C	SIUL2 I/O Pin Multiplexed Signal Configuration Register 0–I/O Pin Multiplexed Signal Configuration Register 511 (SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511)	R/W	32	32/16/8	on page 247
0x0A40–0x123C	SIUL2 Multiplexed Signal Configuration Register for Multiplexed Input Selection 512–Multiplexed Signal Configuration Register for Multiplexed Input Selection 1023 (SIUL2_MSCR_MUX_512–SIUL2_MSCR_MUX_1023)	R/W	32	32/16/8	on page 250
0x1240–0x12FF	Reserved				
0x1300–0x14FF	SIUL2 GPIO Pad Data Out Register 0–GPIO Pad Data Out Register 511 (SIUL2_GPDO0–SIUL2_GPDO511)	R/W	8	32/16/8	on page 251
0x1500–0x16FF	SIUL2 GPIO Pad Data In Register 0–GPIO Pad Data In Register 511 (SIUL2_GPDI0–SIUL2_GPDI511)	R	8	32/16/8	on page 251
0x1700–0x173F	SIUL2 Parallel GPIO Pad Data Out Register 0–Parallel GPIO Pad Data Out Register 31 (SIUL2_PGPDO0–SIUL2_PGPDO31)	R/W	16	32/16/8	on page 252
0x1740–0x177F	SIUL2 Parallel GPIO Pad Data In Register 0–Parallel GPIO Pad Data In Register 31 (SIUL2_PGPDI0–SIUL2_PGPDI31)	R	16	32/16/8	on page 253
0x1780–0x17FF	SIUL2 Masked Parallel GPIO Pad Data Out Register 0–Masked Parallel GPIO Pad Data Out Register 31 (SIUL2_MPGPDO0–SIUL2_MPGPDO31)	R/W	32	32(1)	on page 253

1. 32-bit writes, 8/16/32-bit reads are allowed (only 8-bit and 16-bit writes are not supported and generate a transfer error.)

Note: Reserved registers will be read as 0; writes will have no effect. If supported and enabled in this MCU, a transfer error will be issued when trying to access completely reserved register space. 8/16 for when defined as only 32-bit access gives transfer error. Unimplemented registers accesses give transfer error i.e., due to register not being present

13.2.2 Register descriptions

This section describes in address order all the SIUL2 registers. Each description includes a standard register diagram with an associated figure number.

13.2.2.1 SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

This register contains identification information about the device. The reset values are:

- eTQFP80: 0x5726_1400
- eTQFP100: 0x5726_2400

Address: 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	1	0	0	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PKG					0	0	MAJOR_MASK				MINOR_MASK			
W																
Reset	0	x(1)	x(1)	x(1)	x(1)	x(1)	0	0	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)

Figure 54. SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

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

Table 113. SIUL2_MIDR1 field descriptions

Field	Description
0–15 PARTNUM	MCU Part Number—These digits identify the part number of the chip. Default is 5726.
16	Reserved
17–21 PKG	Package Settings—Can by read by software to determine the package type that is used for the particular device: 0b00101: 80-pin QFP 0b01001: 100-pin QFP
22–23	Reserved
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.

Address: 0x0008 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SF	FLASH_SIZE_1				FLASH_SIZE_2			0	0	0	0	0	0	0	0
W																
Reset	0	0(1)	1(1)	1(1)	0(1)	0(1)	1(1)	0(1)	0(1)	0	0	0	0	0	0	0

	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)	0(1)	1(1)	0(1)	1(1)	1(1)	0	0	0	0	0	0	0	0

Figure 55. SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

1. Values are set at factory and cannot be modified.

Table 114. SIUL2_MIDR2 field description

Field	Description
0 SF	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: 0b000016 KB 0b000132 KB 0b001064 KB 0b0011128 KB 0b0100256 KB 0b0101512 KB 0b01101 MB ... 0b1110256 MB 0b1111512 MB
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: 0b00000 x (FLASH_SIZE_1 / 8) 0b00011 x (FLASH_SIZE_1 / 8) 0b00102 x (FLASH_SIZE_1 / 8) 0b00113 x (FLASH_SIZE_1 / 8) 0b01004 x (FLASH_SIZE_1 / 8) ... 0b111014 x (FLASH_SIZE_1 / 8) 0b111115 x (FLASH_SIZE_1 / 8)
9–15	Reserved

Table 114. SIUL2_MIDR2 field description(Continued)

Field	Description
16–23 FAMILYNUM	ASCII character in MCU Part Number: 0x4CL All other values reserved for future use.
24–31	Reserved

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.

Address: 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	EIF 10	0	0	0	0	EIF5	0	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 56. SIUL2 DMA/Interrupt Status Flag Register (SIUL2_DISR0)

Table 115. SIUL2_DISR0 field descriptions

Field	Description
0–20	Reserved
21 EIF10	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
22–25	Reserved
26 EIF5	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

Table 115. SIUL2_DISR0 field descriptions(Continued)

Field	Description
27	Reserved
28–31 EIF3 EIF2 EIF1 EIF0	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 Register0 (SIUL2_DISR0). The type of request enabled is determined by the corresponding DMA/Interrupt Request Select bit in SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0). This register enables the interrupt messaging to the interrupt controller.

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	EIRE10	0	0	0	0	EIRE5	0	EIRE3	EIRE2	EIRE1	EIRE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 57. SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)

Table 116. SIUL2_DIRER0 field descriptions

Field	Description
0–20	Reserved
21 EIRE10	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
22–25	Reserved
26 EIRE5	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

Table 116. SIUL2_DIRER0 field descriptions(Continued)

Field	Description
27	Reserved
28–31 EIRE3 EIRE2 EIRE1 EIRE0	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 Register0 (SIUL2_DISR0) and SIUL2 DMA/Interrupt Request Enable Register0 (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.

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	0	DIRS10	0	0	0	0	DIRS5	0	DIRS3	DIRS2	DIRS1	DIRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 58. SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)

Table 117. SIUL2_DIRSR0 field descriptions

Field	Description
0–20	Reserved
21 DIRS10	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
22–25	Reserved
26 DIRS5	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

Table 117. SIUL2_DIRSR0 field descriptions(Continued)

Field	Description
27	Reserved
28–31 DIRS3 DIRS2 DIRS1 DIRS0	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.

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	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	IREE10	0	0	0	0	IREE5	0	IREE3	IREE2	IREE1	IREE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 59. SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

Table 118. SIUL2_IREER0 field descriptions

Field	Description
0–20	Reserved
21 IREE10	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
22–25	Reserved
26 IREE5	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
27	Reserved
28–31 IREE3 IREE2 IREE1 IREE0	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.

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	0	0	0	0	0	IFEE10	0	0	0	0	IFEE5	0	IFEE3	IFEE2	IFEE1	IFEE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 60. SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

Table 119. SIUL2_IFEER0 field descriptions

Field	Description
0–20	Reserved
21 IFEE10	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
22–25	Reserved
26 IFEE5	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
27	Reserved
28–31 IFEE3 IFEE2 IFEE1 IFEE0	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.

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	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	IFE10	0	0	0	0	IFE5	0	IFE3	IFE2	IFE1	IFE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 61. SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

Table 120. SIUL2_IFER0 field descriptions

Field	Description
0–20	Reserved
21 IFE10	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
22–25	Reserved
26 IFE5	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
27	Reserved
28–31 IFE3 IFE2 IFE1 IFE0	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.

Address: 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	MAXCNT0			
W																													[3:0]			
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 62. SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0)

Table 121. SIUL2_IFMCR0 field descriptions

Field	Description
28–31 MAXCNTx	Maximum Interrupt Filter Counter setting MAXCNTx can be 0 to 15 For MAXCNTx = 0, 1, 2, the filter behaves as ALL PASS filter For MAXCNTx = 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. T(DELAY) = (1 to 5)*TCK + MAXCOUNT * TCK. The SW is to expect 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.*Section 21.6: Peripheral clocks*

Address: 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[3:0]			
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 63. 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 MSCRs control the I/O pin function and electrical properties for each pin on the device, and are used to select the input signal for IP blocks on the device that have multiple input sources. There is a dedicated MSCR for each I/O pin on the device. Each MSCR is

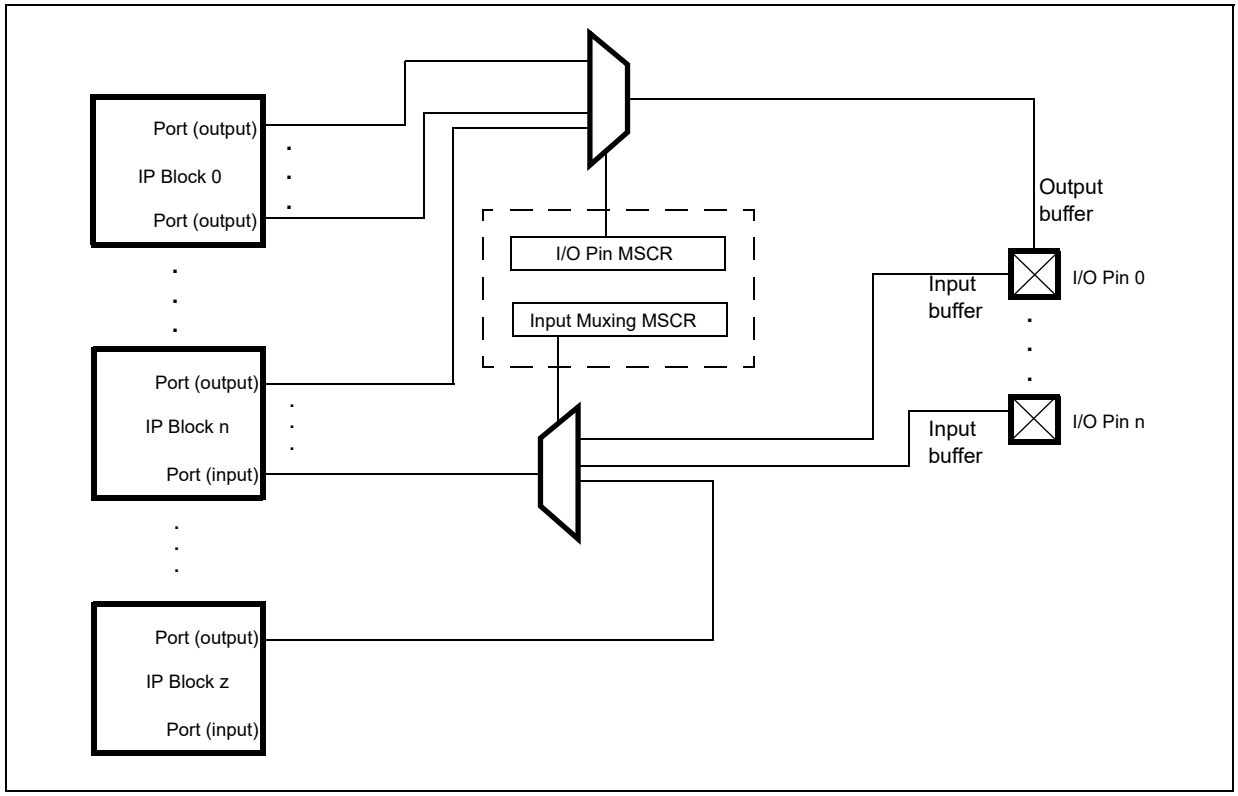
independent of the others, except in the case of an LVDS configuration, where two MSCRs must be configured together in this mode.

The MSCRs are divided into two sets:

- I/O pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511)
- Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR_MUX_512–SIUL2_MSCR_MUX_1023)

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 64*.

Figure 64. MSCR I/O pin and IP block port connectivity



13.2.2.11.1 I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511)

The I/O pin MSCR definition is given in *Figure 65*. 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.

Address: 0x0240–0x0A3C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	OERC		0	ODC			SMC	APC	ILS		IBE	HYS	WPDE	WPUE
W																
Reset	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	INV								SSS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 65. I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511)

Table 123. SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511 field description

Field	Description[1]
0–1	Reserved
2–3 OERC	Output Edge Rate Control—Used only when the associated destination is a chip pin. Specifies the driver strength of the associated pin. 00b Weak 01b Medium 10b Strong 11b Very strong **Note:** Refer to the SPC572Lx I/O Signal Description and Input Multiplexing Tables
4–5	Reserved
5–7 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. 000 Output buffer disabled 001 Open-drain 010 Push-pull 011 Open-source 100 Microsecond Channel LVDS 101 LFAST LVDS 110–111 Reserved
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)
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. 00b Automotive 01b TTL 10b LVDS 11b CMOS

Table 123. SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511 field description(Continued)

Field	Description[1]
12 IBE	Input Buffer Enable—Used only when the associated destination is a chip pin. Enables the associated pin's input buffer. 0 Disabled 1 Enabled
13 HYS	Input Hysteresis—Used only when the associated destination is a chip pin. Enables input hysteresis for the associated pin. 0 Disabled 1 Enabled
14 WPDE	Weak Pulldown Enable—Used only when the associated destination is a chip pin. Enables the associated pin's weak pulldown resistor. It is OK for both WPDE and WPUE to be enabled. 0 Disabled 1 Enabled
15 WPUE	Weak Pullup Enable—Used only when the associated destination is a chip pin. Enables the associated pin's weak pullup resistor. It is OK for both WPDE and WPUE to be enabled. 0 Disabled 1 Enabled
16 INV	Invert—Inverts the signal selected by SSS before transmitting it to the associated destination (chip pin or module port). 0 Don't invert 1 Invert
17–23	Reserved
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 13.2.2.11.3: Chip-pin MSCR assignments and related information* and *Section 13.2.2.11.4: Module-port MSCR assignments and SSS values*.

1. The SPC572Lx I/O Signal Description and Input Multiplexing Tables are contained in a Microsoft Excel® workbook file attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

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 *Figure 65* applies to all GPIO pins on the devices, which defaults to input/pull-up enabled. The exceptions to this are documented in *Table 124*. Only the SMC, HYS, WPDE, and WPUE bits have exceptions to the default MSCR reset value.

Table 124. I/O MSCR reset state exceptions

Port	Critical function	SMC	HYS	SSS	WPDE	WPUE
PA[4]	NMI	1	0	—	1	0
PA[5]	JCOMP	1	1	0x0001	1	0
PA[6]	TCK	1	1	0x0001	1	0

Table 124. I/O MSCR reset state exceptions(Continued)

Port	Critical function	SMC	HYS	SSS	WPDE	WPUE
PA[7]	TMS	1	1	0x0001	1	0
PA[8]	TDI	1	1	0x0001	0	1
PA[9]	TDO	1	1	0x0001	0	0
PA[14]	—	1	1		0	1
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

13.2.2.11.2 Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR_MUX_512–SIUL2_MSCR_MUX_1023)

The multiplexed input selection MSCR definition is given in *Figure 66*. 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).

Address: 0X0A40–0x123C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV								SSS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 66. Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR_MUX_512–SIUL2_MSCR_MUX_1023)

Table 125. SIUL2_MSCR_MUX_512–SIUL2_MSCR_MUX_1023 field description

Field	Description(1)
0–15	Reserved
16 INV	Invert—Inverts the signal selected by SSS before transmitting it to the associated destination (chip pin or module port). 0 Don't invert 1 Invert

Table 125. SIUL2_MSCR_MUX_512–SIUL2_MSCR_MUX_1023 field description(Continued)

Field	Description(1)
17–23	Reserved
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 13.2.2.11.3: Chip-pin MSCR assignments and related information* and *Section 13.2.2.11.4: Module-port MSCR assignments and SSS values*.

1. The SPC572Lx I/O Signal Description and Input Multiplexing Tables are contained in a Microsoft Excel® workbook file attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

13.2.2.11.3 Chip-pin MSCR assignments and related information

For chip-pin MSCR assignments, pin types, APC support, and SSS values, see the signal description table in the “Signal description” chapter.

13.2.2.11.4 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* shows the assigned MSCR, and the SSS values and their corresponding source signals.

13.2.2.12 SIUL2 GPIO Pad Data Output Registers (SIUL2_GPDO0–SIUL2_GPDO511)

These registers can be used to set or clear a single GPIO pad with a byte access. See Signal Description chapter for mapping of ports to GPDO registers.

Address: 0x1300–0x14FF Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	PDO [x]
W								
Reset	0	0	0	0	0	0	0	0

Figure 67. SIUL2 GPIO Pad Data Output registers (SIUL2_GPDO0–SIUL2_GPDO511)

Table 126. SIUL2_GPDO0–SIUL2_GPDO511 field description

Field	Description
7 PDO[x]	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

13.2.2.13 SIUL2 GPIO Pad Data Input Registers (SIUL2_GPDI0–SIUL2_GPDI511)

These registers can be used to read the GPIO pad data with a byte access. See Signal Description chapter for mapping of ports to GPDI registers.

Address: 0x1500–0x16FF Access: User read

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	PDI [x]
W								
Reset	0	0	0	0	0	0	0	0

Figure 68. SIUL2 GPIO Pad Data Input Registers (SIUL2_GPDI0–SIUL2_GPDI511)

Table 127. SIUL2_GPDI0–SIUL2_GPDI511 field description

Field	Description
7 PDI[x]	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

13.2.2.14 SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO0_1–SIUL2_PGPDO30_31)

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.

Address: 0x1700–0x173F 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	PPDO[x][0:15]				PPDO[x+1][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 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO0)

Table 128. SIUL2_PGPDO0 field description

Field	Description
0–15 PPDO[x]	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 are coherent with access to the bit-wise GPIO Pad Data Output Registers (SIUL2_GPDO). The x and y bit index define which PPDO register bit is equivalent to which PDO register bit according to the following equation: PPDO[x][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:

PPDO[0][0] = PDO[0]

PPDO[2][0] = PDO[32]

PPDO[31][15] = PDO[511]

13.2.2.15 SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI0_1–SIUL2_PGPDI30_31)

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: 0x1740–0x177F Access: User read-only

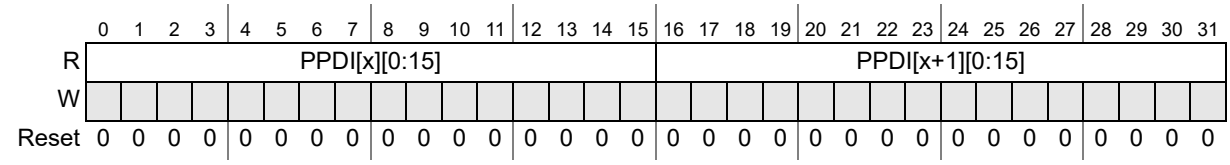


Figure 70. SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI0)

Table 129. SIUL2_PGPDI0 field description

Field	Description
0–15 PPDI[x]	Parallel Pad Data In—Reads the current pad value. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Input Registers (SIUL2_GPDI). The x and y bit index define which PPDI register bit is equivalent to which PDI register bit according to the following equation: PPDI[x][y] = PDI[(x*16)+y]

13.2.2.16 SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO0–SIUL2_MPGPDO31)

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.

Address: 0x1780–0x17FF Access: User read/write

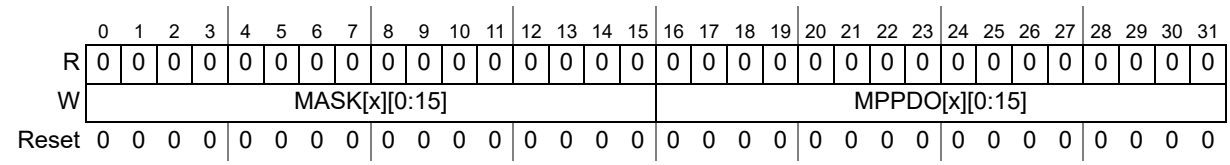


Figure 71. SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO0)

Table 130. SIUL2_MPGPDO0 field description

Field	Description
0–15 MASK[x] [0:15]	Mask Field—Each bit corresponds to one data bit in the MPPDO[x] register at the same bit location. 0 The associated bit value in the MPPDO[x] field is ignored 1 The associated bit value in the MPPDO[x] field is written
16–31 MPPDO[x] [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 are coherent with access to the bit-wise GPIO Pad Data Output Registers (PDO). The x and bit index define which MPPDO register bit is equivalent to which PDO register bit according to the following equation: MPPDO[x][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 85 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.
  - Up to 4 different drive strength levels can be supported.
- 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)

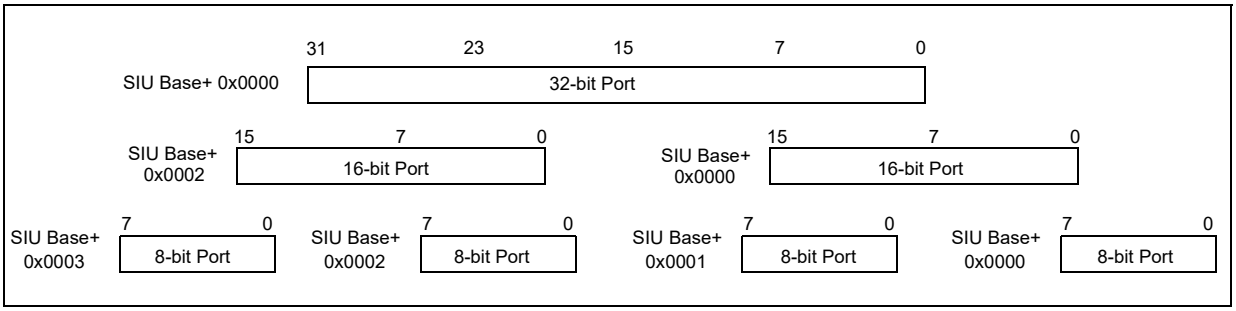
The SIUL2 allows each pad to be configured as either a General Purpose Input Output pad (GPIO), and as one or more alternate functions (input or output), the function of which is determined by the peripheral that will use the pad.

GPIO pads can also optionally be implemented without any alternate function.

The SIUL2 is designed to manage 92 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 72*, all port accesses are identical with each read or write being performed only at a different location to access a different port width.

Figure 72. Data Port example arrangement showing configuration for different port width accesses



This implementation requires that the registers are arranged in such a way as to support this range of port widths without having to split reads or writes into multiple accesses.

The SIUL2 has separate data input and data output registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

The data output registers support both read and write operations to be performed.

The data input registers support read access only.

When the pad is configured to use one of its alternate functions, the data input value reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non GPIO), this write will not be reflected by the pad value until reconfigured to GPIO.

All general purpose pads are implemented as bidirectional.

Note: In case the bidirectional operation impacts some performances (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”.

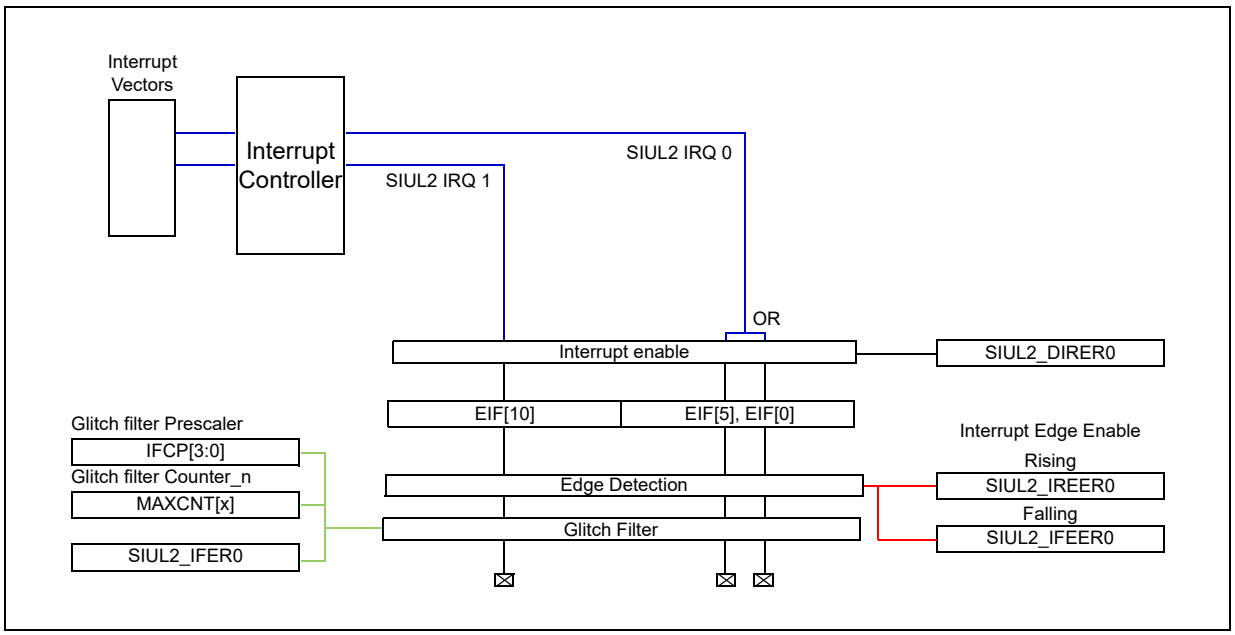
13.3.4 External interrupts/DMA requests (EIRQ pins)

The SIUL2 supports three external interrupts which can be allocated to any pad necessary at the MCU level. This allocation is fixed per MCU.

The SIUL2 supports two interrupt vectors to the interrupt controller of the MCU. Each interrupt vector can support three external interrupt sources from the device pads.

Refer to *Figure 73* for an overview of the external interrupt implementation.

Figure 73. External interrupt pad diagram



All the external interrupt pads within a single group (max 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.

Note: External interrupt pins should never be configured as outputs (MSCR_ODC bits are not zeros) 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 EIREEn bits in DIREER0.

13.3.4.2 External interrupt management

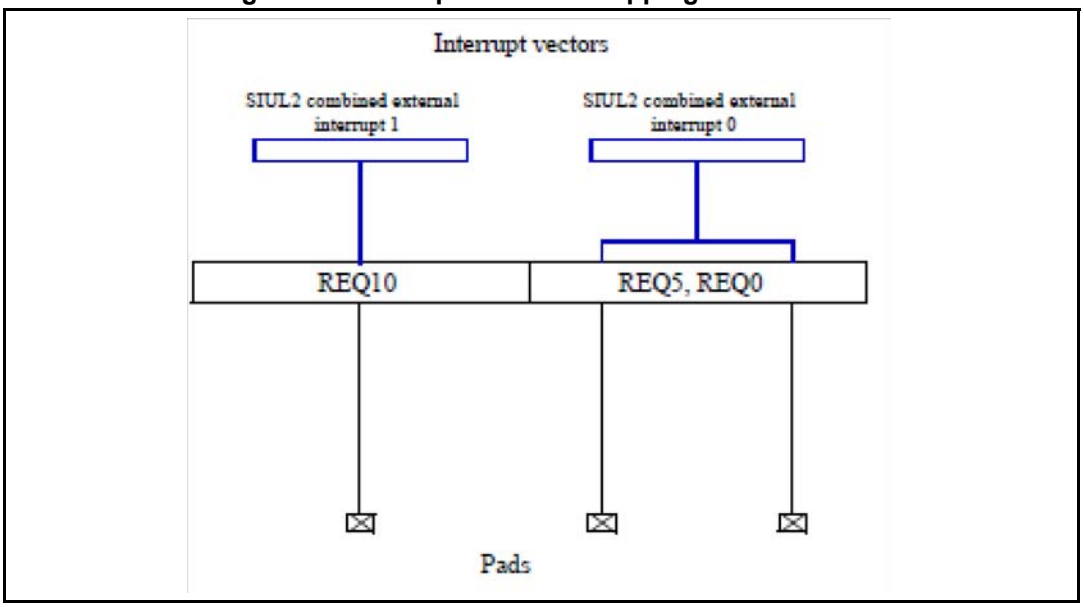
Each interrupt can be enabled or disabled independently. This can be performed using a single rolled up register (SIUL2_DIRER0 – *Figure 57*). A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges 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 SPC572Lx device supports two SIUL2 interrupt vectors aggregating a total of three external interrupt sources as shown in *Figure 74*.

Figure 74. Interrupt to vector mapping at MCU level



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 including a 4-beat, 64-bit burst for cache line accesses
- 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 Chip Configuration details 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 131. XBAR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0	XBAR Priority Register Slave 0 (XBAR_PRS0)	32	R	See Chip Config	*on page 260*
10	XBAR Control Register 0 (XBAR_CRS0)	32	R/W	0000_0000h	*on page 263*
100	XBAR Priority Register Slave 1 (XBAR_PRS1)	32	R	See Chip Config	*on page 260*
110	XBAR Control Register 1 (XBAR_CRS1)	32	R/W	0000_0000h	*on page 263*
200	XBAR Priority Register Slave 2 (XBAR_PRS2)	32	R	See Chip Config	*on page 260*
210	XBAR Control Register 2 (XBAR_CRS2)	32	R/W	0000_0000h	*on page 263*
300	XBAR Priority Register Slave 3 (XBAR_PRS3)	32	R	See Chip Config	*on page 260*
310	XBAR Control Register 3 (XBAR_CRS3)	32	R/W	0000_0000h	*on page 263*
400	XBAR Priority Register Slave 4 (XBAR_PRS4)	32	R	See Chip Config	*on page 260*
410	XBAR Control Register 4 (XBAR_CRS4)	32	R/W	0000_0000h	*on page 263*
500	XBAR Priority Register Slave 5 (XBAR_PRS5)	32	R	See Chip Config	*on page 260*
510	XBAR Control Register 5 (XBAR_CRS5)	32	R/W	0000_0000h	*on page 263*
600	XBAR Priority Register Slave 6 (XBAR_PRS6)	32	R	See Chip Config	*on page 260*
610	XBAR Control Register 6 (XBAR_CRS6)	32	R/W	0000_0000h	*on page 263*
700	XBAR Priority Register Slave 7 (XBAR_PRS7)	32	R	See Chip Config	*on page 260*
710	XBAR Control Register 7 (XBAR_CRS7)	32	R/W	0000_0000h	*on page 263*

14.1.2.1 XBAR Priority Registers Slave (XBAR_PRS*n*)

The priority registers (XBAR_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_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 *Section 6.3.2: Crossbar switch configuration* for XBAR_PRS*n* priority values.

Offset: 0x0000 + n*0x100 (n = 0 to 7) 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	See Chip Configuration section for this device's reset values.															

	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	See Chip Configuration section for this device's reset values.															

Figure 75. Priority Registers Slave (XBAR_PRS*n*)

Table 132. XBAR_PRS*n* field descriptions

Field	Description
0 Reserved	This read-only bit is reserved and always has the value zero.
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.
4 Reserved	This read-only bit is reserved and always has the value zero.
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.
8 Reserved	This read-only bit is reserved and always has the value zero.
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.

Table 132. XBAR_PRS*n* field descriptions(Continued)

Field	Description
12 Reserved	This read-only bit is reserved and always has the value zero.
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.
16 Reserved	This read-only bit is reserved and always has the value zero.
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.
20 Reserved	This read-only bit is reserved and always has the value zero.
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.
24 Reserved	This read-only bit is reserved and always has the value zero.
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.

Table 132. XBAR_PRS*n* field descriptions(Continued)

Field	Description
28 Reserved	This read-only bit is reserved and always has the value zero.
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 7) 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	HPEx							
W																
Reset	0	0	0	0	0	0	0	0	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	ARB		0		PCTL		0	PARK		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 76. XBAR Control Register (XBAR_CRS*n*)

Table 133. 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.
2–7 Reserved	This read-only bitfield is reserved and always has the value zero.

Table 133. XBAR_CRS*n* field descriptions(Continued)

Field	Description
8–15 HPEx	High Priority Enable Determines if Master x is able to temporarily elevate it's request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master x are ignored and are treated the same as regular requests. 1 High Priority requests from Master x can temporarily elevate the priority level of the master's request to the slave.
16–21 Reserved	This read-only bitfield is reserved and always has the value zero.
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
24–25 Reserved	This read-only bitfield is reserved and always has the value zero.
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
28 Reserved	This read-only bit is reserved and always has the value zero.
29–31 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. 000 Park on master port M0 001 Park on master port M1 010 Park on master port M2 011 Park on master port M3 100 Park on master port M4 101 Park on master port M5 110 Park on master port M6 111 Park on master port M7

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_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_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_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 or a locked 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 locked and 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.

14.2.3.3 Priority assignment

Each master port needs to be assigned a unique 3-bit priority level. If an attempt is made to program multiple master ports with the same priority level within the priority registers (XBAR_PRS*n*), the crossbar switch responds with a bus error and the registers are not updated.

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 Peripheral Bridge

15.1 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 6: 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.

15.1.1 Features

Key features of the peripheral bridge are:

- Supports a pair of 32-bit transactions for selected 64-bit memory accesses
- Each independently configurable peripheral includes a clock enable, which allows peripherals to operate at any speed less than the system clock rate
- Programming model that provides memory protection functionality

15.1.2 General operation

The slave devices connected to the peripheral bridge are modules that contain readable/writable control and status registers. The system masters read and write these registers through the peripheral bridge. The peripheral bridge generates module enables, the module address, transfer attributes, byte enables, and write data as inputs to the peripherals. The peripheral bridge captures read data from the peripheral interface and drives it to the crossbar switch.

Each peripheral is typically allocated one 16-KB block (or slot) of the memory map. Usually, the peripheral bridge uses the size of the addressed peripheral to perform proper data byte lane routing only; no bus decomposition (dynamic bus sizing) is performed. However, there are the following exceptions:

- Aligned 64-bit reads are permitted to 32-bit slots.
- Aligned 64-bit writes may be targeted to 32-bit slots.
- Aligned or misaligned 64-bit instruction fetches may also be performed to 32-bit slots.

All other 64-bit accesses result in an error response.

15.2 Memory map and register definition

Table 134. Peripheral bridge memory map

Offset address	Register	Width (bits)	Location
0x000	Master Privilege Register A (AIPS_MPRA)	32	*on page 269*
0x004	**Reserved**	—	—
0x100	Peripheral Access Control Register A (AIPS_PACRA)	32	*on page 270*

Table 134. Peripheral bridge memory map(Continued)

Offset address	Register	Width (bits)	Location
0x104	Peripheral Access Control Register B (AIPS_PACRB)	32	on page 270
0x108	Peripheral Access Control Register C (AIPS_PACRC)	32	on page 270
0x10C	Peripheral Access Control Register D (AIPS_PACRD)	32	on page 270
0x110	**Reserved**	—	—
0x114	Peripheral Access Control Register F (AIPS_PACRF)	32	on page 270
0x118	Peripheral Access Control Register G (AIPS_PACRG)	32	on page 270
0x11C	Peripheral Access Control Register H (AIPS_PACRH)	32	on page 270

15.2.1 Master Privilege Registers (AIPS_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 6: Device configuration* for details about the master assignments to these registers.

Address: 0x000 Access: Supervisor 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	MPROT0	MPROT1	MPROT2	MPROT3	MPROT4	MPROT5	MPROT6	MPROT7
Reset	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1

Figure 77. Master Privilege Register A (MPRA)

The MPROT*n* field is defined as shown in *Figure 78*.

	0	1	2	3
R	0	MTR	MTW	MPL
W				

Figure 78. MPROT*n* fields

Table 135. MPROT*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 MTR	Master trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.

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

Field	Description
2 MTW	Master trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL	Master privilege level Determines how the privilege level of the master is determined. 0 Accesses from this master are forced to user mode. 1 Accesses from this master are not forced to user mode.

15.2.2 Peripheral Access Control Registers (AIPS_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 136 shows the top-level structure of the PACRs.

Table 136. PACR memory map

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0100	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x0104	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x0108	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x010C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31
0x0110	Reserved								
0x0114	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47
0x0118	PACRG	PACR48	PACR49	PACR50	PACR51	PACR52	PACR53	PACR54	PACR55
0x011C	PACRH	PACR56	PACR57	PACR58	PACR59	PACR60	PACR61	PACR62	PACR63

Address: 0x100–11C (AIPS_PACRA–H) Access: Supervisor 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	PACR*n*	PACR(*n*+1)	PACR(*n*+2)	PACR(n+3)	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	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0

Note: The reset value of PACR0 is 0101b.

Figure 79. Peripheral Access Control Registers (AIPS_PACRA–H)

Each PACR*n* field is defined as shown in *Figure 80*.

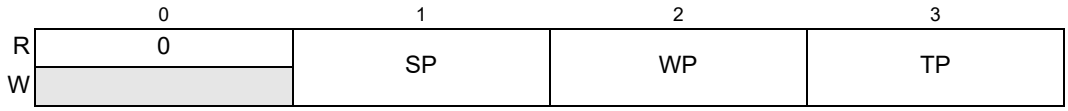


Figure 80. PACR*n* fields

Table 137. 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.

15.2.3 Off-Platform Peripheral Access Control Registers (AIPS_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 138 shows the top-level structure of the OPACR registers.

Table 138. 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	OPACR8	OPACR 9	OPACR 10	OPACR 11	OPACR 12	OPACR 13	OPACR 14	OPACR 15
0x0148	OPACRC	OPACR 16	OPACR 17	OPACR 18	OPACR 19	OPACR 20	OPACR 21	OPACR 22	OPACR 23

Table 138. OPACR memory map(Continued)

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
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
0x0194	OPACRV	OPACR 168	OPACR 169	OPACR 170	OPACR 171	OPACR 172	OPACR 173	OPACR 174	OPACR 175

Table 138. OPACR memory map(Continued)

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

Address: 0x140–1BC Access: Supervisor 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	OPACR*n*	OPACR (*n*+1)	OPACR (*n*+2)	OPACR (*n*+3)	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	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0

Figure 81. Off-Platform Peripheral Access Control Registers (AIPS_OPACRA–AF)

The OPACR*n* field is defined as in *Figure 82*.

	0	1	2	3
R	0	SP	WP	TP
W				

Figure 82. OPACR*n* fields

Table 139. 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.
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.

15.3 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.

15.3.1 Access support

Aligned and misaligned 32-bit and 16-bit accesses, as well as byte accesses, are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the slave peripheral bus. Peripheral registers must not be misaligned, although no explicit checking is performed by the peripheral bridge. The peripheral bridge performs two slave peripheral bus transfers for allowed 64-bit transactions, to 32-bit sized peripheral slots only. All other accesses are performed with a single transfer.

All accesses to the peripheral slots must be sized less than or equal to the designated peripheral slot size. If an access is attempted which is larger (in size) than the targeted port, an error response is generated without a peripheral access.

16 System Memory Protection Unit (SMPU)

16.1 Overview

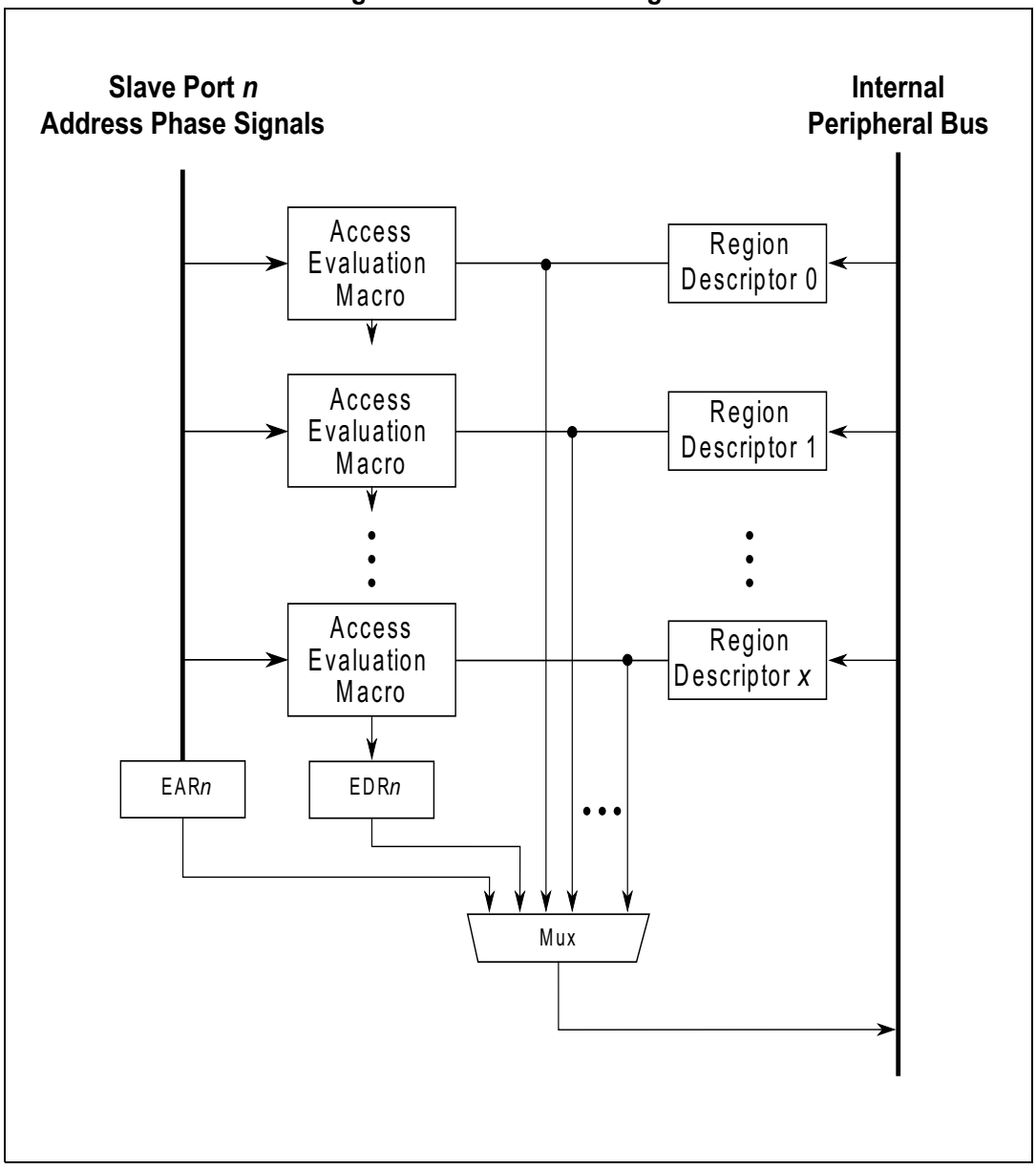
The System Memory Protection Unit (SMPU) provides hardware access control for system bus memory references. The SMPU concurrently monitors and evaluates system bus transactions using pre-programmed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

16.2 Block diagram

A simplified block diagram of the SMPU module is shown in the following figure. The hardware's two-dimensional connection matrix is clearly visible with the basic access evaluation macro shown as the replicated submodule block. The crossbar switch slave ports are shown on the left, the region descriptor registers are in the middle, and the peripheral bus interface is on the right side. The evaluation macro contains two magnitude comparators connected to the start and end address registers from each region descriptor as well as the combinational logic blocks to determine the region hit and detect protection violations.

For details of the access evaluation macro, see *Section 16.5.1*.

Figure 83. SMPU block diagram



16.3 Features

The SMPU feature set includes:

- Supports up to 24 program-visible 128-bit region descriptors, accessible as four 32-bit words each. (Specific module instances may support fewer than 24).
  - Each region descriptor defines an arbitrarily sized space, aligned anywhere in memory
    - Region sizes can vary from a minimum of 1 byte to a maximum of (4 GB minus 1 byte)
  - Read/write access control permissions defined in region descriptor
  - Cache-inhibit attribute indicator per region descriptor
  - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues
  - Priority given to granting permission over denying access for overlapping region descriptors
- Supports as many as eight crossbar slave ports
- Detects access errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the SMPU inhibits the bus cycle being sent to the targeted slave device.
- Aborted core accesses generate instruction storage (ISI) or data storage (DSI) interrupts (see the documentation related to the core for details).
- Error registers (per bus master ID) capture the last faulting address, attributes, and other information.
- Global SMPU enable/disable control bit

16.4 Memory map and register definition

16.4.1 Memory map

The programming model is partitioned into three groups: control registers, error capture registers, and the data structure containing the region descriptors.

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

The programming model can be accessed only in supervisor mode.

Note: See the device configuration details for any chip-specific register information for this module. For example, a specific instance of a module may support only up to eight region descriptors.

Table 140. SMPU*x* memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section
FC01_0000	SMPU0_CESR0	32	R/W	0000_8002h	*Section 16.4.2.1*
FC01_0004	SMPU0_CESR1	32	R/W	0000_8003h	*Section 16.4.2.2*
FC01_0100	SMPU0_EAR0	32	R	0000_0000h	*Section 16.4.2.3*
FC01_0104	SMPU0_EDR0	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0108	SMPU0_EAR1	32	R	0000_0000h	*Section 16.4.2.3*
FC01_010C	SMPU0_EDR1	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0110	SMPU0_EAR2	32	R	0000_0000h	*Section 16.4.2.3*
FC01_0114	SMPU0_EDR2	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0118	SMPU0_EAR3	32	R	0000_0000h	*Section 16.4.2.3*
FC01_011C	SMPU0_EDR3	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0120	SMPU0_EAR4	32	R	0000_0000h	*Section 16.4.2.3*
FC01_0124	SMPU0_EDR4	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0128	SMPU0_EAR5	32	R	0000_0000h	*Section 16.4.2.3*
FC01_012C	SMPU0_EDR5	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0130	SMPU0_EAR6	32	R	0000_0000h	*Section 16.4.2.3*
FC01_0134	SMPU0_EDR6	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0138	SMPU0_EAR7	32	R	0000_0000h	*Section 16.4.2.3*
FC01_013C	SMPU0_EDR7	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0140	SMPU0_EAR8	32	R	0000_0000h	*Section 16.4.2.3*
FC01_0144	SMPU0_EDR8	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0148	SMPU0_EAR9	32	R	0000_0000h	*Section 16.4.2.3*
FC01_014C	SMPU0_EDR9	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0150	SMPU0_EAR10	32	R	0000_0000h	*Section 16.4.2.3*
FC01_0154	SMPU0_EDR10	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0158	SMPU0_EAR11	32	R	0000_0000h	*Section 16.4.2.3*
FC01_015C	SMPU0_EDR11	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0160	SMPU0_EAR12	32	R	0000_0000h	*Section 16.4.2.3*
FC01_0164	SMPU0_EDR12	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0168	SMPU0_EAR13	32	R	0000_0000h	*Section 16.4.2.3*
FC01_016C	SMPU0_EDR13	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0170	SMPU0_EAR14	32	R	0000_0000h	*Section 16.4.2.3*
FC01_0174	SMPU0_EDR14	32	R	0000_0080h	*Section 16.4.2.4*
FC01_0178	SMPU0_EAR15	32	R	0000_0000h	*Section 16.4.2.3*
FC01_017C	SMPU0_EDR15	32	R	0000_0080h	*Section 16.4.2.4*

Table 140. SMPU*x* memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section
FC01_0400	SMPU0_RGD0_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_0404	SMPU0_RGD0_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_0408	SMPU0_RGD0_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_0408	SMPU0_RGD0_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*
FC01_040C	SMPU0_RGD0_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*
FC01_0410	SMPU0_RGD1_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_0414	SMPU0_RGD1_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_0418	SMPU0_RGD1_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_0418	SMPU0_RGD1_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*
FC01_041C	SMPU0_RGD1_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*
FC01_0420	SMPU0_RGD2_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_0424	SMPU0_RGD2_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_0428	SMPU0_RGD2_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_0428	SMPU0_RGD2_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*
FC01_042C	SMPU0_RGD2_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*
FC01_0430	SMPU0_RGD3_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_0434	SMPU0_RGD3_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_0438	SMPU0_RGD3_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_0438	SMPU0_RGD3_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*
FC01_043C	SMPU0_RGD3_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*
FC01_0440	SMPU0_RGD4_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_0444	SMPU0_RGD4_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_0448	SMPU0_RGD4_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_0448	SMPU0_RGD4_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*
FC01_044C	SMPU0_RGD4_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*
FC01_0450	SMPU0_RGD5_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_0454	SMPU0_RGD5_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_0458	SMPU0_RGD5_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_0458	SMPU0_RGD5_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*
FC01_045C	SMPU0_RGD5_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*
FC01_0460	SMPU0_RGD6_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_0464	SMPU0_RGD6_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_0468	SMPU0_RGD6_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_0468	SMPU0_RGD6_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*

Table 140. SMPU*x* memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section
FC01_046C	SMPU0_RGD6_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*
FC01_0470	SMPU0_RGD7_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_0474	SMPU0_RGD7_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_0478	SMPU0_RGD7_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_0478	SMPU0_RGD7_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*
FC01_047C	SMPU0_RGD7_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*
FC01_0480	SMPU0_RGD8_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_0484	SMPU0_RGD8_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_0488	SMPU0_RGD8_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_0488	SMPU0_RGD8_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*
FC01_048C	SMPU0_RGD8_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*
FC01_0490	SMPU0_RGD9_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_0494	SMPU0_RGD9_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_0498	SMPU0_RGD9_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_0498	SMPU0_RGD9_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*
FC01_049C	SMPU0_RGD9_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*
FC01_04A0	SMPU0_RGD10_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_04A4	SMPU0_RGD10_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_04A8	SMPU0_RGD10_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_04A8	SMPU0_RGD10_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*
FC01_04AC	SMPU0_RGD10_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*
FC01_04B0	SMPU0_RGD11_WORD0	32	R/W	0000_0000h	*Section 16.4.2.5*
FC01_04B4	SMPU0_RGD11_WORD1	32	R/W	0000_0000h	*Section 16.4.2.6*
FC01_04B8	SMPU0_RGD11_WORD2_FMT0(1)	32	R/W	0000_0000h	*Section 16.4.2.7*
FC01_04B8	SMPU0_RGD11_WORD2_FMT1(1)	32	R/W	0000_0000h	*Section 16.4.2.8*
FC01_04BC	SMPU0_RGD11_WORD3	32	R/W	0000_0000h	*Section 16.4.2.9*

1. SMPU0_RGDn_WORD2_FMT0 implemented when SMPU0_RGDn_WORD3[FMT] = 0. SMPU0_RGDn_WORD2_FMT1 implemented when SMPU0_RGDn_WORD3[FMT] = 1.

16.4.2 Register descriptions

16.4.2.1 Control/Error Status Register 0 (SMPU0_CESR0)

Address : SMPU0_CESR0 – FC01_0000h base + 0h offset = FC01_0000h Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MERR															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0											HRL			GVLD
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 84. Control/Error Status Register 0 (SMPU0_CESR0)

Table 141. SMPU0_CESR0 field descriptions

Field	Description
0–15 MERR	Master *n* error, where the bus master number matches the bit number Indicates a captured error in EAR*n* and EDR*n*. A bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing ‘1’ to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction (or equivalent) can detect the presence of a captured error. 0 No error has occurred for bus master *n*. 1 An error has occurred for bus master *n*.
16 Reserved	This read-only bit is reserved and always has the value ‘1’.
17–27 Reserved	This read-only bitfield is reserved and always has the value ‘0’.
28–30 HRL	Hardware revision level Specifies the SMPU’s hardware and definition revision level. It can be read by software to determine the functional definition of the module.
31 GVLD	Global Valid (global enable/disable for the SMPU) 0 SMPU is disabled. All accesses from all bus masters are allowed. 1 SMPU is enabled

16.4.2.2 Control/Error Status Register 1 (SMPU0_CESR1)

Address SMPU0_CESR1 – FC01_0000h base + 4h offset = FC01_0004h Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MEOVR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0											NRGD			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X

Figure 85. Control/Error Status Register 1 (SMPU0_CESR1)

Table 142. SMPU0_CESR1 field descriptions

Field	Description
0–15 MEOVR	Master *n* error overrun, where the bus master number matches the bit number Each bit in this field signals that another SMPU error for bus master *n* has occurred before the previous error was processed. The details of the first error are recorded in the EAR*n* and EDR*n* registers, and no information on subsequent errors is recorded until the associated CESR0[MERR] flag is cleared. 0 No error overrun condition has been detected for bus master *n*. 1 An error overrun condition has been detected for bus master *n*.
16 Reserved	This read-only bit is reserved and always has the value ‘1’.
17–27 Reserved	This read-only bitfield is reserved and always has the value ‘0’.
28–31 NRGD	Number of region descriptors Indicates the number of region descriptors implemented in the SMPU. 0001 4 region descriptors 0010 8 region descriptors 0011 12 region descriptors 0100 16 region descriptors 0101 20 region descriptors 0110 24 region descriptors

16.4.2.3 Error Address Register, Bus Master *n* (SMPU0_EAR*n*)

When the SMPU detects an access error on bus master *n*, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR0[MERR] is set. Additional information about the faulting access is captured in the corresponding EDR*n* at the same time.

Note: The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the original error is processed.

Address SMPU0_EAR*n* – FC01_0100h + (8d × n), where *n* = 0d to 15d Access: Supervisor read-only

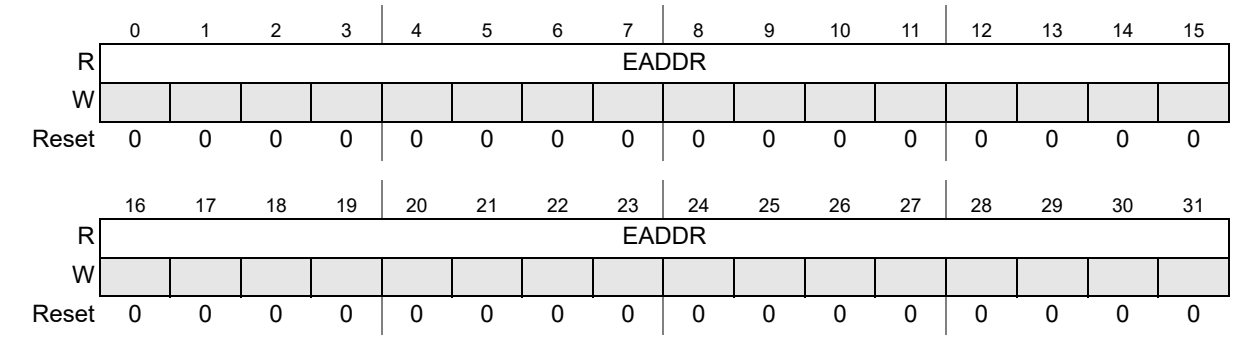


Figure 86. Error Address Register, Bus Master *n* (SMPU0_EAR*n*)

Table 143. SMPU0_EARn field descriptions

Field	Description
0–31 EADDR	Error address Indicates the reference address from bus master n that generated the access error

16.4.2.4 Error Detail Register, Bus Master *n* (SMPU0_EDR*n*)

When the SMPU detects an access error associated with bus master *n*, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR0[MERR] is set. Information on the faulting address is captured in the corresponding EAR*n* register at the same time.

Note: The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the original error is processed.

Address SMPU0_EDR*n* – FC01_0104h + (8d × n), where *n* = 0d to 15d Access: Supervisor read-only

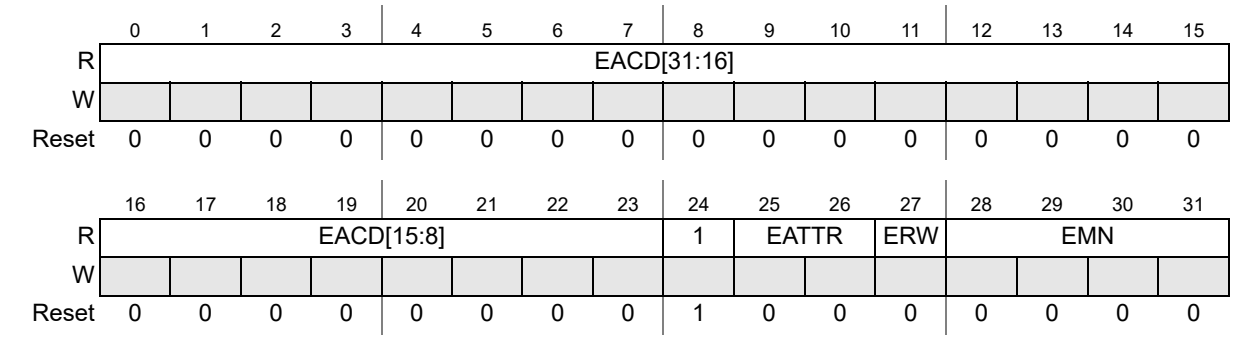


Figure 87. Error Detail Register, Bus Master *n* (SMPU0_EDR*n*)

Table 144. SMPU0_EDR*n* field descriptions

Field	Description
0–23 EACD	Error access control detail Indicates the region descriptor with the access error, where the region descriptor number matches the bit number. If EDR*n* contains a captured error and EACD is all zeroes, an access did not hit in any region descriptor. If only a single EACD bit is set, the access error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, the access error was caused by an overlapping set of region descriptors.
24 Reserved	This read-only bit is reserved and always has the value ‘1’.
25–26 EATTR	Error attributes Indicates attribute information about the faulting reference. 00 User mode, instruction access 01 User mode, data access 10 Supervisor mode, instruction access 11 Supervisor mode, data access
27 ERW	Error read/write Indicates the access type of the faulting reference. 0 Read 1 Write
28–31 EMN	Error master number Indicates the logical bus master number of the faulting reference. This field is used to determine the bus master that generated the access error.

16.4.2.5 Region Descriptor *n*, Word 0 (SMPU0_RGD*n*_WORD0)

Address: SMPU0_RGD*n*_WORD0 – FC01_0400h + (16 × *n*), where *n* = 0 to 11 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SRTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SRTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 88. Region Descriptor *n*, Word 0 (SMPU0_RGD*n*_WORD0)

Table 145. SMPU0_RGD*n*_WORD0 field descriptions

Field	Description
0–31 SRTADDR	Start address Defines the byte start address of the memory region.

16.4.2.6 Region Descriptor *n*, Word 1 (SMPU0_RGD*n*_WORD1)

Address: SMPU0_RGD*n*_WORD1 – FC01_0404h + (16 × *n*), where *n* = 0 to 11 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ENDADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ENDADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 89. Region Descriptor *n*, Word 1 (SMPU0_RGD*n*_WORD1)

Table 146. SMPU0_RGD*n*_WORD1 field descriptions

Field	Description
0–31 ENDADDR	End address Defines the byte end address of the memory region. **Note:** The SMPU does not verify that ENDADDR ≥ SRTADDR.

16.4.2.7 Region Descriptor *n*, Word 2 Format 0 (SMPU0_RGD_WORD2_FMT0)

RGD_WORD2 has two formats as determined by the RGD_WORD3[FMT] field.

Note: RGD_WORD2_FMT0 applies when RGD_WORD3[FMT] = 0.

RGD_WORD2_FMT0 defines the access control rights of the memory region on a per master basis. The access control rights are defined by separate read and write permissions. For these fields, the bus master number refers to the logical bus master number.

For the access control rights, there are two flags per logical bus master:

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

Each field consists of the two flags, with (r) being in the more significant position. For example, M0P has (r) as bit 0 and (w) as bit 1.

The bit settings are as follows:

- If set, the corresponding flag allows the specific access type (r = memory read, w = memory write).
- If cleared, the specific access type is not allowed.

Writes to this word clear the region descriptor's valid bit.

Address: SMPU0_RGD*n*_WORD2_FMT0 – FC01_0408h + (16 × *n*), where *n* = 0 to 11

Access: Supervisor read/write

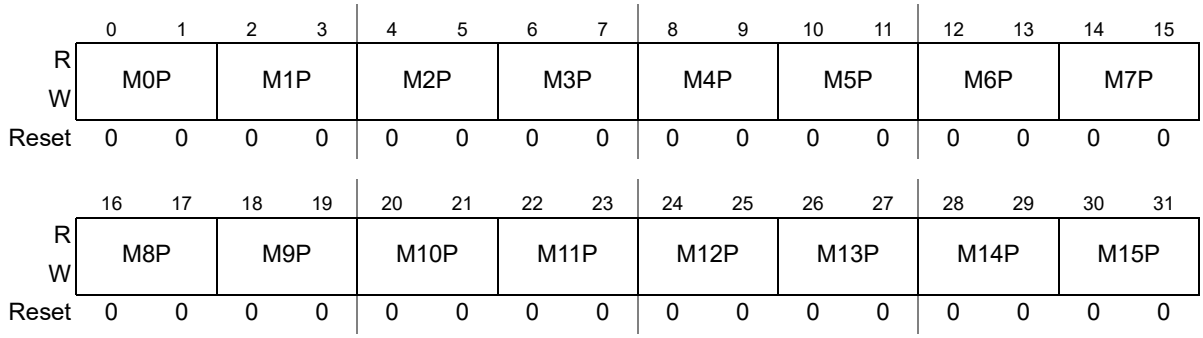


Figure 90. Region Descriptor *n*, Word 2 Format 0 (SMPU0_RGD*n*_WORD2_FMT0)

Table 147. SMPU0_RGD*n*_WORD2_FMT0 field descriptions

Field	Description
0–1 M0P	Bus master 0 permissions
2–3 M1P	Bus master 1 permissions
4–5 M2P	Bus master 2 permissions
6–7 M3P	Bus master 3 permissions
8–9 M4P	Bus master 4 permissions
10–11 M5P	Bus master 5 permissions
12–13 M6P	Bus master 6 permissions
14–15 M7P	Bus master 7 permissions
16–17 M8P	Bus master 8 permissions
18–19 M9P	Bus master 9 permissions
20–21 M10P	Bus master 10 permissions
22–23 M11P	Bus master 11 permissions
24–25 M12P	Bus master 12 permissions

Table 147. SMPU0_RGD*n*_WORD2_FMT0 field descriptions(Continued)

Field	Description
26–27 M13P	Bus master 13 permissions
28–29 M14P	Bus master 14 permissions
30–31 M15P	Bus master 15 permissions

16.4.2.8 Region Descriptor *n*, Word 2 Format 1 (SMPU0_RGD*n*_WORD2_FMT1)

RGD_WORD2 has two formats as determined by the RGD_WORD3[FMT] field.

Note: RGD_WORD2_FMT1 applies when RGD_WORD3[FMT] = 1.

Note: A specific module instance of the SMPU may support only the FMT0 format. If so, the FMT field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies.

RGD_WORD2_FMT1 defines pointers for each master to select access control flags contained in Word3. For these fields, the bus master number refers to the *logical* bus master number.

The bit settings are as follows:

- 00: RGD*n* cannot be accessed by this master, that is, SR = SW = SX = UR = UW = UX = 0 (where “S” is for supervisor permissions and “U” is for user permissions).
- 01: RGD*n* access flags are defined in RGD*n*_WORD3[ACCSET1].
- 10: RGD*n* access flags are defined in RGD*n*_WORD3[ACCSET2].
- 11: RGD*n* access flags are defined in RGD*n*_WORD3[ACCSET3].

Writes to this word clear the region descriptor’s valid bit.

Address: SMPU0_RGD*n*_WORD2_FMT1 – FC01_0408h + (16 × *n*), where *n* = 0 to 11 — Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0S		M1S		M2S		M3S		M4S		M5S		M6S		M7S	
Reset	0	0	0	0	0	0	0	0	0	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	M8S		M9S		M10S		M11S		M12S		M13S		M14S		M15S	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 91. Region Descriptor n, Word 2 Format 1 (SMPU0_RGDn_WORD2_FMT1)

Table 148. SMPU0_RGD*n*_WORD2_FMT1 field descriptions

Field	Description
0–1 M0S	Bus master 0 permission select
2–3 M1S	Bus master 1 permission select
4–5 M2S	Bus master 2 permission select
6–7 M3S	Bus master 3 permission select
8–9 M4S	Bus master 4 permission select
10–11 M5S	Bus master 5 permission select
12–13 M6S	Bus master 6 permission select
14–15 M7S	Bus master 7 permission select
16–17 M8S	Bus master 8 permission select
18–19 M9S	Bus master 9 permission select
20–21 M10S	Bus master 10 permission select
22–23 M11S	Bus master 11 permission select
24–25 M12S	Bus master 12 permission select
26–27 M13S	Bus master 13 permission select
28–29 M14S	Bus master 14 permission select
30–31 M15S	Bus master 15 permission select

16.4.2.9 Region Descriptor *n*, Word 3 (SMPU0_RGD*n*_WORD3)

The final word of the SMPU region descriptor contains the valid bit, the format select (FMT), three sets of access permissions flags used when FMT = 1, plus other configuration bits.

For access permissions used when FMT = 1, there are three flags {read, write, execute} for each operating mode (supervisor, user). 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.

Each ACCSET field consists of six independent bits, enabling supervisor read, write, and execute permissions {SR, SW, SX} and user read, write, and execute permissions {UR, UW, UX}. 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 is not performed.

The ACCSET bits are in this order: {SR, SW, SX, UR, UW, UX} with SR being in the most significant position. For example, the SR bit within the ACCSET1 field is register bit 0.

Note: When FMT = 0, the ACCSET fields are not writable.

Address: SMPU0_RGD*n*_WORD3 – FC01_040Ch + (16 × *n*), where *n* = 0 to 11 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ACCSET1						ACCSET2						ACCSET3			
Reset	0	0	0	0	0	0	0	0	0	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	ACCSET3		0									FMT	RO	0	CI	VLD
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 92. Region Descriptor *n*, Word 3 (SMPU0_RGD*n*_WORD3

Table 149. SMPU0_RGD*n*_WORD3 field descriptions

Field	Description
0–5 ACCSET1	Access controls, Set 1 Defines a set of access control options for RGD_WORD2_FMT1.
6–11 ACCSET2	Access controls, Set 2 Defines a set of access control options for RGD_WORD2_FMT1.
12–17 ACCSET3	Access controls, Set 3 Defines a set of access control options for RGD_WORD2_FMT1.

Table 149. SMPU0_RGD*n*_WORD3 field descriptions(Continued)

Field	Description
18–26 Reserved	This read-only bitfield is reserved and always has the value ‘0’.
27 FMT	Region Descriptor Format This bit selects the configuration format (FMT0 or FMT1) for this region descriptor. **Note:** A specific module instance of the SMPU may support only the FMT0 format. If so, the FMT field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies. 0 Use format 0 (RGD_WORD2_FMT0) 1 Use format 1 (RGD_WORD2_FMT1)
28 RO	Read-Only This bit is intended to prevent accidental writes of an SMPU region descriptor. **Note:** Setting RO in an RGD locks all four words of the RGD until a system reset; the valid bit of the RGD and the global valid bit have no effect. 0 The region descriptor can be read or written. 1 Attempted writes to any location in the region descriptor are ignored with an error-free data transfer termination.
29 Reserved	This read-only bit is reserved and always has the value ‘0’.
30 CI	Cache Inhibit Defines the cacheability attribute of the memory region. This bit is returned to the bus master that initiated the memory reference. **Note:** An address range specified in an SMPU region descriptor for a cacheable space (that is, CI = 0) must be defined with a starting address aligned on a 0-modulo-32 byte address and with a multiple of the 32 byte cache line size. 0 References to this region can be cached. 1 References to this region cannot be cached.
31 VLD	Valid Signals the region descriptor is valid. Any write to RGD*n*_WORD0–2 clears this bit. 0 Region descriptor is invalid 1 Region descriptor is valid

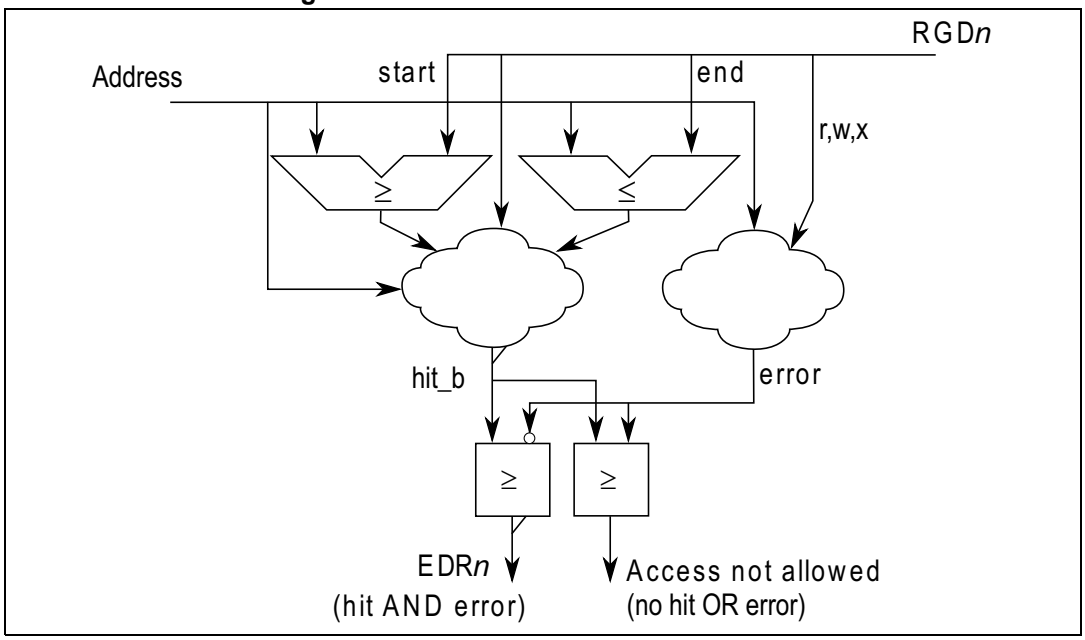
16.5 Functional description

In this section, the functional operation of the SMPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

16.5.1 Access evaluation macro

The basic operation of the SMPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in *Figure 93*, the access evaluation macro inputs the crossbar bus address phase signals and the contents of a region descriptor (RGD*n*), then performs two major functions: region hit determination and detection of an access protection violation. *Figure 93* shows a functional block diagram.

Figure 93. SMPU access evaluation macro



16.5.1.1 Hit determination

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

```
region_hit = ((addr[0:31] >= RGDn_Word0[SRTADDR]) &
      (addr[0:31] <= RGDn_Word1[ENDADDR])) &
      RGDn_Word3[VLD]
```

where addr is the current reference address, RGDn_Word0[SRTADDR] and RGDn_Word1[ENDADDR] are the start and end addresses, and RGDn_Word3[VLD] is the valid bit.

Note: The SMPU does not verify that ENDADDR ≥ SRTADDR.

Note: Region hit determination is based only on an address comparison of the first byte being accessed; that is, the SMPU does not check the size of the access to make sure it fits within an allowed region.

16.5.1.2 Privilege violation determination

While the access evaluation macro is determining the region hit, the logic is also evaluating whether the current access is allowed by the permissions defined in the region descriptor. Using the master signal, a set of effective permissions is generated from the relevant fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions, using the specification shown in *Table 150*.

Table 150. Protection violation definition

Access type	Access permissions		Protection violation?
	r	w	
Instruction fetch read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data write	—	0	Yes, no write permission
	—	1	No, access is allowed

16.5.2 Putting it all together and error terminations

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

1. The access does not hit in any region descriptor.
2. The access hits in a single region descriptor and that region has a protection violation.
3. The access hits in multiple (overlapping) regions and all regions have protection violations.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see *Section 16.7: Application information*.

16.6 Initialization information

At system startup, load the required number of region descriptors, including setting RGD*n*_Word3[VLD]. Setting CESR0[GVLD] enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire SMPU is disabled (CESR0[GVLD] = 0). Next, load the appropriate region descriptors (RGD*n*), including the setting of the RGDn_Word3[VLD] bits. Finally, set CESR0[GVLD] to enable the entire module.

A region descriptor must be set to allow access to the SMPU registers if further changes are needed.

16.7 Application information

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

- Creating a new memory region—Load the appropriate region descriptor into an available RGD*n*, using four sequential 32-bit writes. The hardware assists in the

maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.

- Modifying a region descriptor—Load the updates into the region descriptor using sequential 32-bit writes. Writing to Word3 re-enables the region descriptor valid bit. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.
- Removing a region descriptor—Clearing RGD*n*_Word3[VLD] deletes an existing region descriptor.
- Accessing the SMPU—Allocate a region descriptor to restrict SMPU access to supervisor mode from a specific master.
- Detecting an access error—The current bus cycle is terminated with an error response and EAR*n* and EDR*n* capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R*n*. CESR0[MERR] signals. The error registers contain the captured fault data.
- Overlapping region descriptors—Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

The following dual-core system example contains four bus masters: the two processors (CP0, CP1) and two DMA engines (DMA1, a traditional data movement engine transferring data between RAM and peripherals, and DMA2, a second engine transferring data to/from the RAM only). Consider the following region descriptor assignments:

Table 151. Overlapping region descriptor example

Region description	RGDn		CP0	CP1	DMA1	DMA2	System memory map space
CP0 code	0		rwx	r--	—	—	Flash
CP1 code	1		r--	rwx	—	—	
CP0 data & stack	2		rw-	—	—	—	RAM
CP0 → CP1 shared data	2	3	r--	r--	—	—	
CP1 → CP0 shared data	4						
CP1 data & stack	4		—	rw-	—	—	
Shared DMA data	5		rw-	rw-	rw	rw	
SMPU	6		rw-	rw-	—	—	Peripheral space
Peripherals	7		rw-	rw-	rw	—	

In this example, there are eight descriptors used to span nine regions in the three main spaces of the system memory map (flash, RAM, and peripheral space). Each region indicates the specific permissions for each of the four bus masters, and this definition provides an assigned set of shared, private, and executable memory spaces.

Of particular interest are the two overlapping spaces: region descriptors 2 & 3 and 3 & 4.

The space defined by RGD2 with no overlap is a private data and stack area that provides read/write access to CP0 only. The overlapping space between RGD2 and RGD3 defines a shared data space for passing data from CP0 to CP1, and the access controls are defined by the logical OR of the two region descriptors. Thus, CP0 has (rw- | r--) = (rw-) permissions, while CP1 has (--- | r--) = (r--) permission in this space. Both DMA engines are excluded from this shared processor data region. The overlapping spaces between RGD3 and RGD4 defines another shared data space, this one for passing data from CP1 to CP0. For this overlapping space, CP0 has (r-- | ---) = (r--) permission, while CP1 has (rw- | r--) = (rw-) permission. The non-overlapped space of RGD4 defines a private data and stack area for CP1 only.

The space defined by RGD5 is a shared data region, accessible by all four bus masters. Finally, the slave peripheral space mapped onto the peripheral bus is partitioned into two regions: one containing the SMPU's programming model accessible only to the two processor cores and the remaining peripheral region accessible to both processors and the traditional DMA1 master.

This simple example is intended to show one possible application of the capabilities of the SMPU in a typical system.

17 Intelligent AHB Gasket (IAHBG)

17.1 Introduction

This section details the intelligent bridging gasket between the fast and slow clock domains. The intelligent operation of the gasket initiates pending reads early and optimizes bursts to recover some of the performance that is lost on a registered interface between different clock domains. Pending reads begin even if there is not an hready to the master. Non-bursted writes only begin when the slave is IDLE, pending writes wait until IDLE.

The IAHBG complies with the AHBV6lite protocol, including an extension to support masters that do address retraction during the error termination sequence or abort a burst read due to an error termination or another non-error termination like cache inhibit termination.

Table 152 lists the available operation modes.

Table 152. Operation modes available

Operating mode (Master:Slave)	Comments
0:0	Flowthru (no register wall)
1:1	Master and slave at same operating frequency; gasket performance optimizations on by default
2:1	Master at twice frequency of slave; gasket performance optimizations on by default
1:2	Slave at twice frequency of master; gasket performance optimizations on by default
4:1	Master at four times frequency of slave; gasket performance optimizations disabled at this time
1:4	Slave at four times frequency of master; gasket performance optimizations disabled at this time

17.2 Timing diagrams

The following are a sample of timing diagrams assuming a zero wait state response. The 0:0 flowthru mode is not shown.

17.2.1 1:1

In the 1:1 timing mode the master is running at the same clock frequency as the slave. There are different read slots that the pending read transaction can begin executing early, but only the first slot is shown. The pending read can either be a single or burst transaction.

Figure 94. Read, pending read (1:1 timing mode)

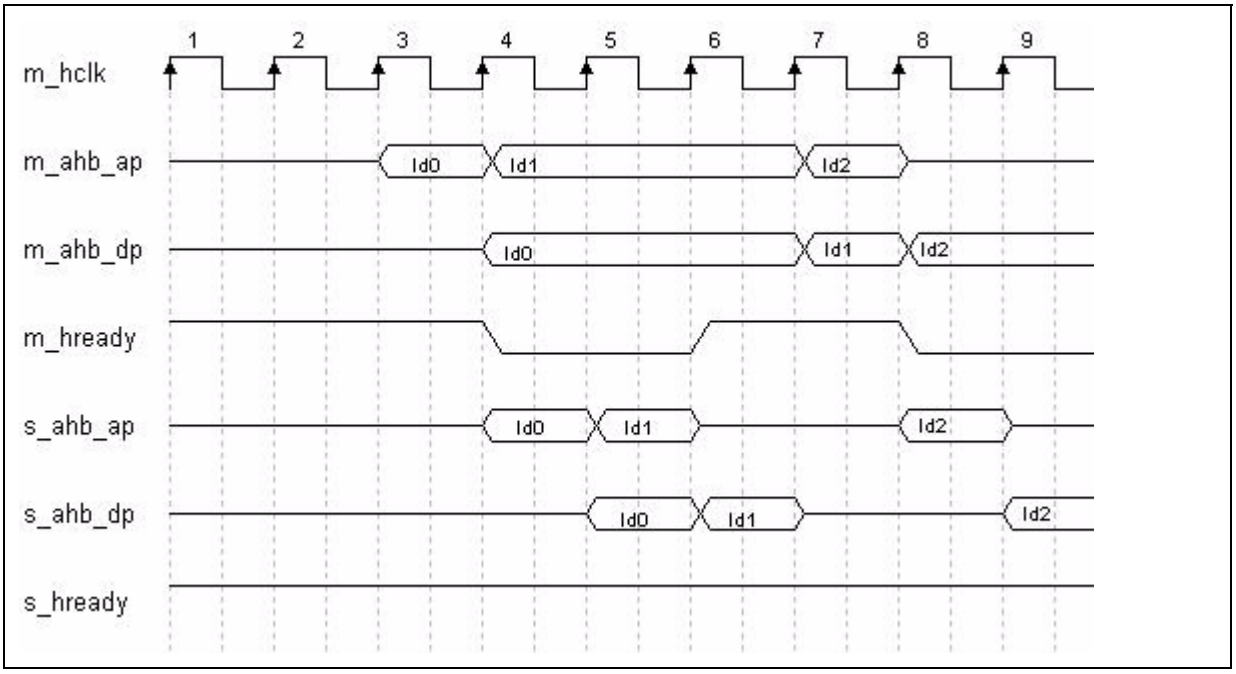


Figure 95. Write, pending read (1:1 timing mode)

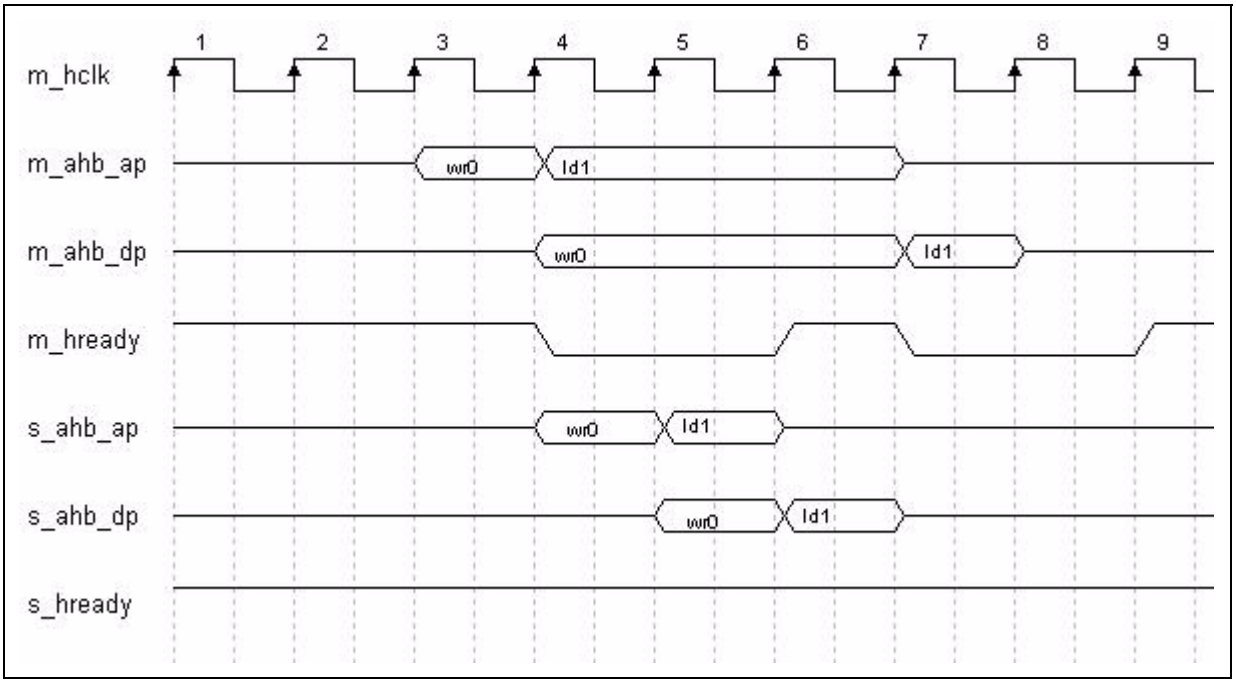


Figure 96. Burst4 read (1:1 timing mode)

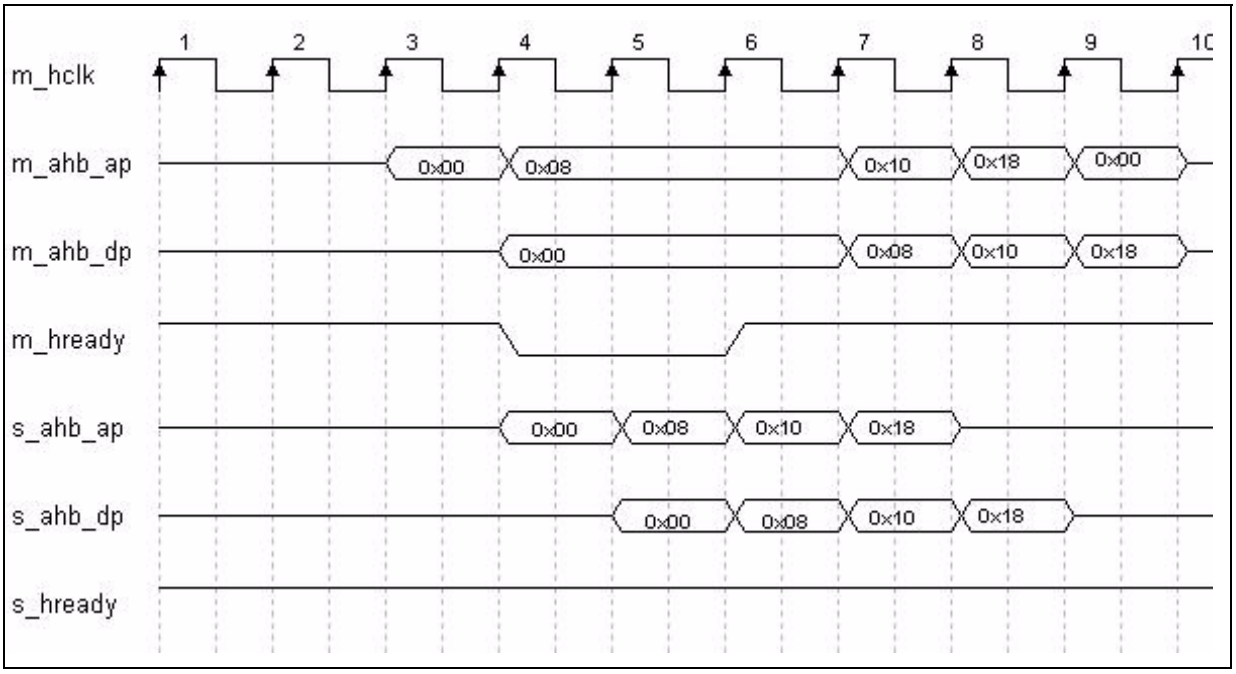
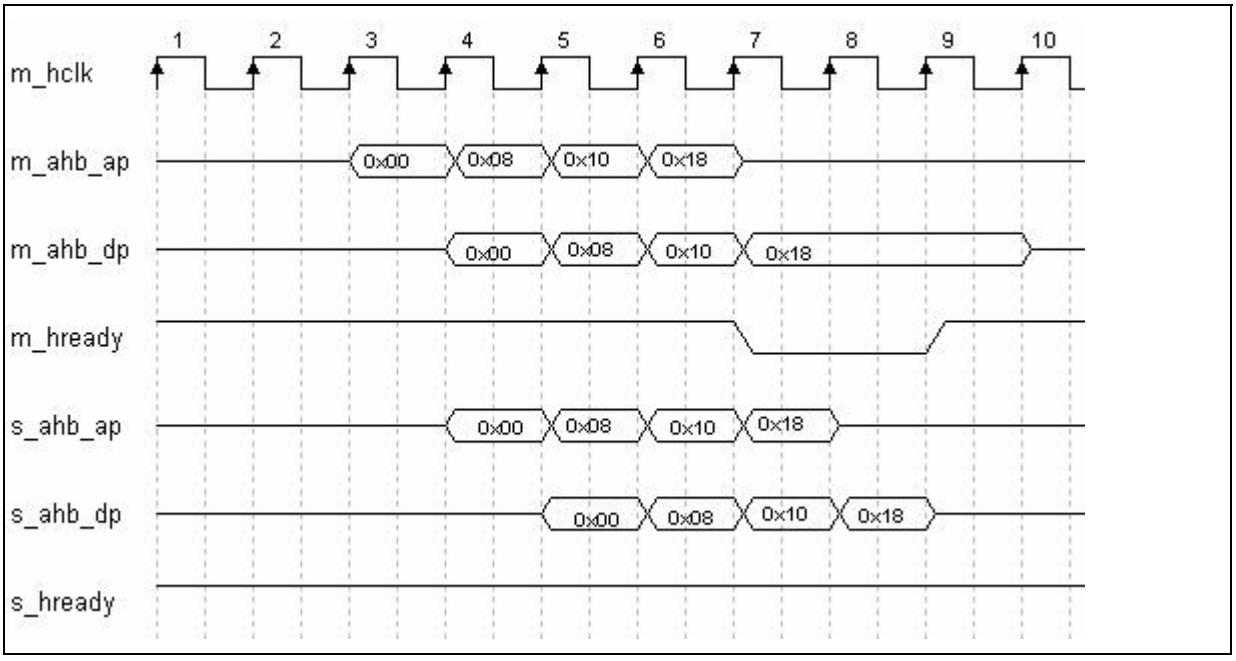


Figure 97. Burst4 write (1:1 timing mode)



17.2.2 2:1

In the 2:1 timing mode the master is running at twice the clock frequency of the slave. The 4:1 mode is not shown because it is very similar to the 2:1 mode. There are different read slots that the pending read transaction can begin executing early, but only the first slot is shown. The pending read can either be a single or burst transaction.

Figure 98. Read, pending read (2:1 timing mode)

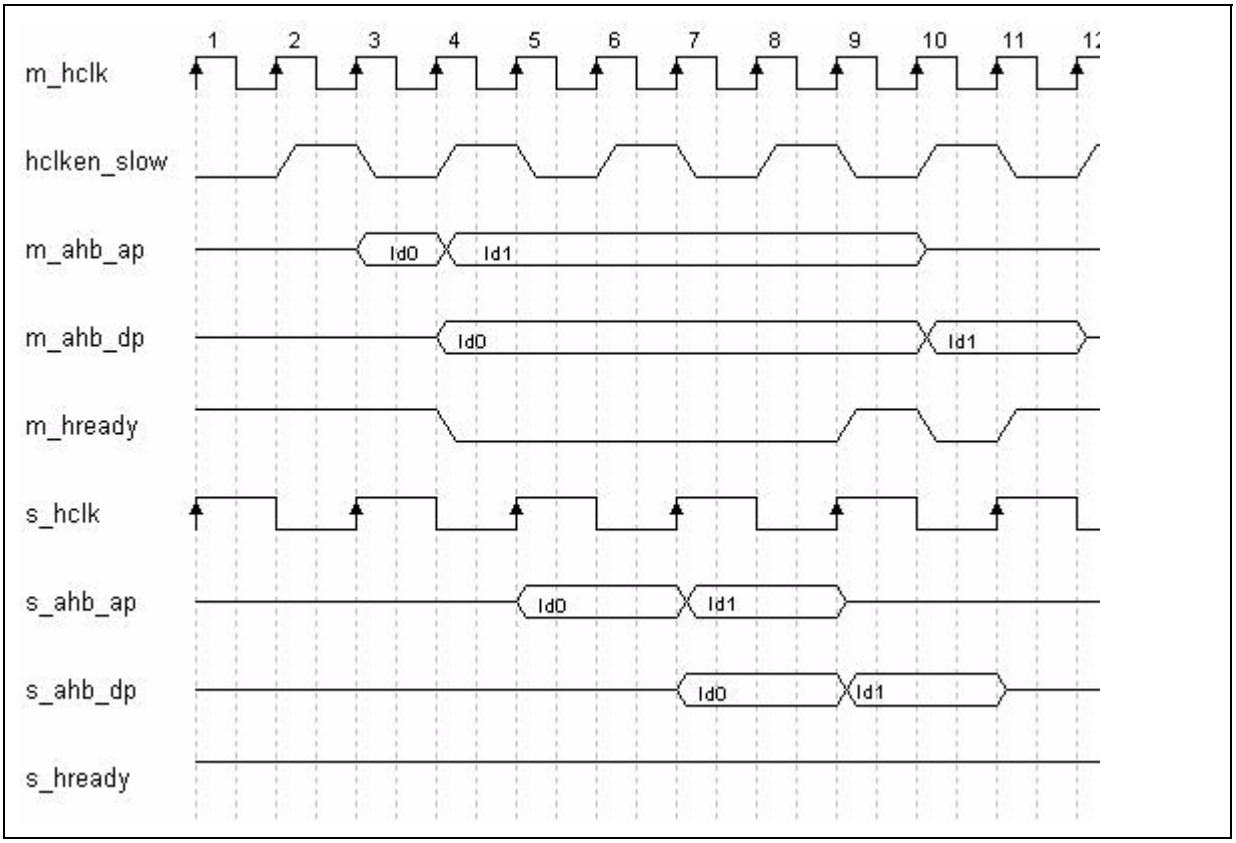


Figure 99. Write, pending read (2:1 timing mode)

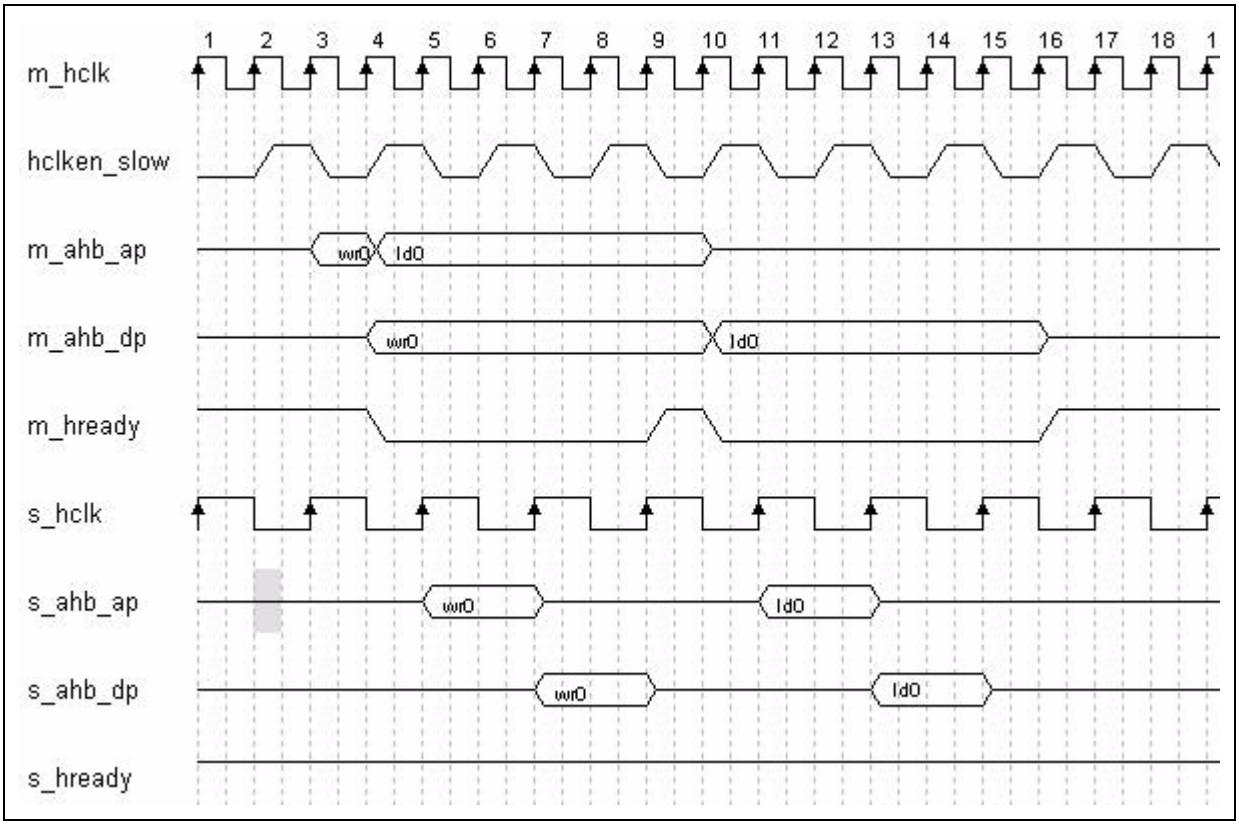


Figure 100. Burst4 read (2:1 timing mode)

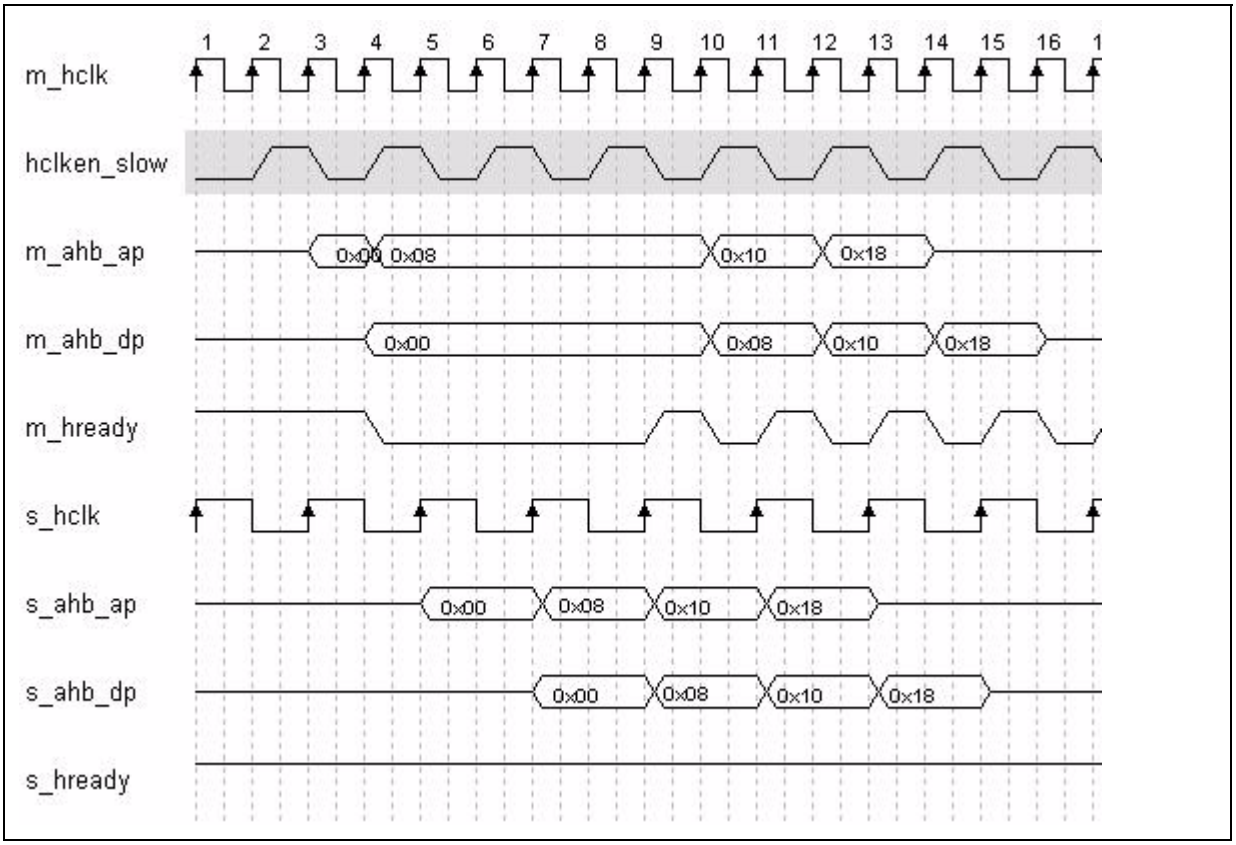
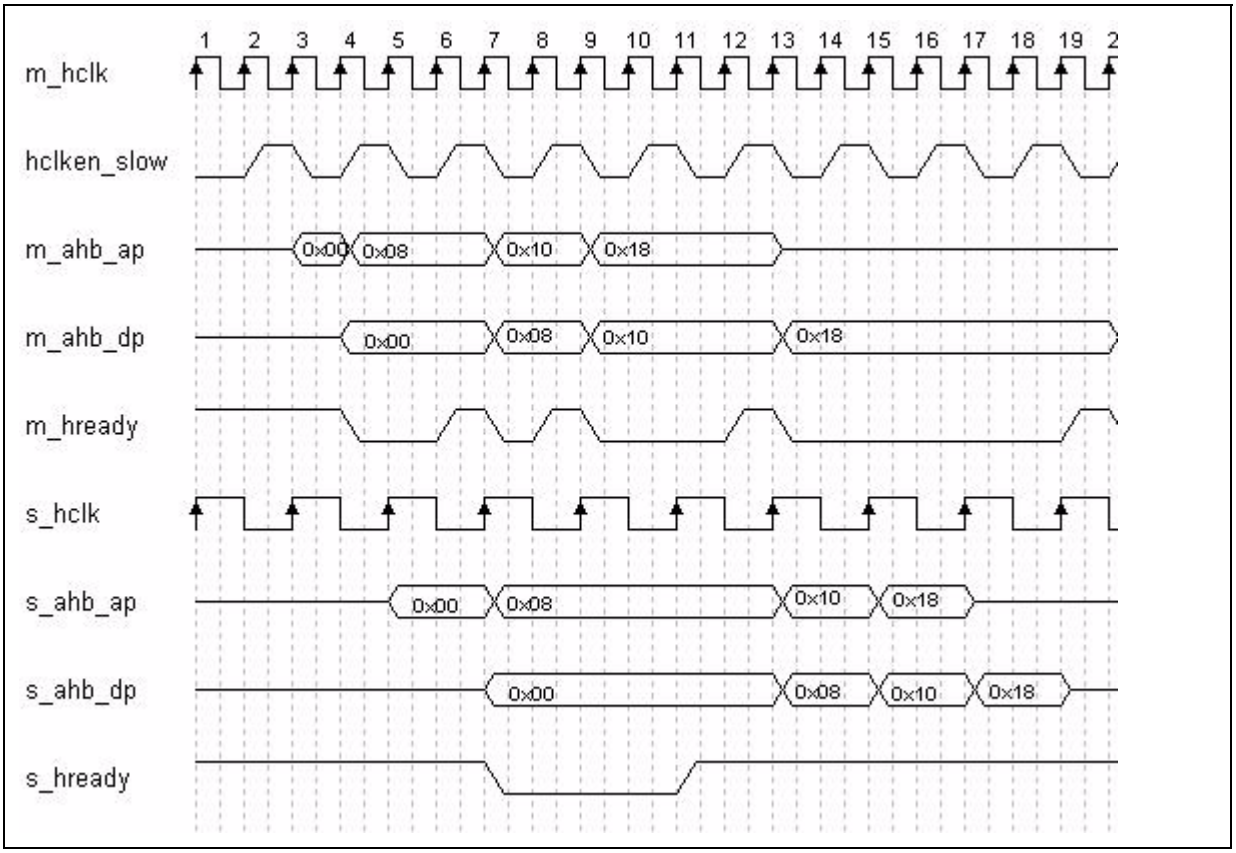


Figure 101. Burst4 write (2:1 timing mode)



17.2.3 1:2

In the 1:2 timing mode, the slave is running at twice the clock frequency of the master. The 1:4 mode is not shown because it is very similar to the 1:2 mode. There are different read slots that the pending read transaction can begin executing early, but only the first slot is shown. The pending read can either be a single or burst transaction.

Because the slave is operating at a faster clock frequency than the master, the transactions from the master must be stored into a queue of data and response attribute registers. At a certain point during the master burst transactions, the queue is either emptied into the master for a burst read or it is emptied into the slave for a burst write. Burst writes will only work across the IAHBG if an assumption is made where the hresp from the slave is accumulated into the last beat of the burst (implicit burst write mode).

Figure 102. Read, pending read (1:2 timing mode)

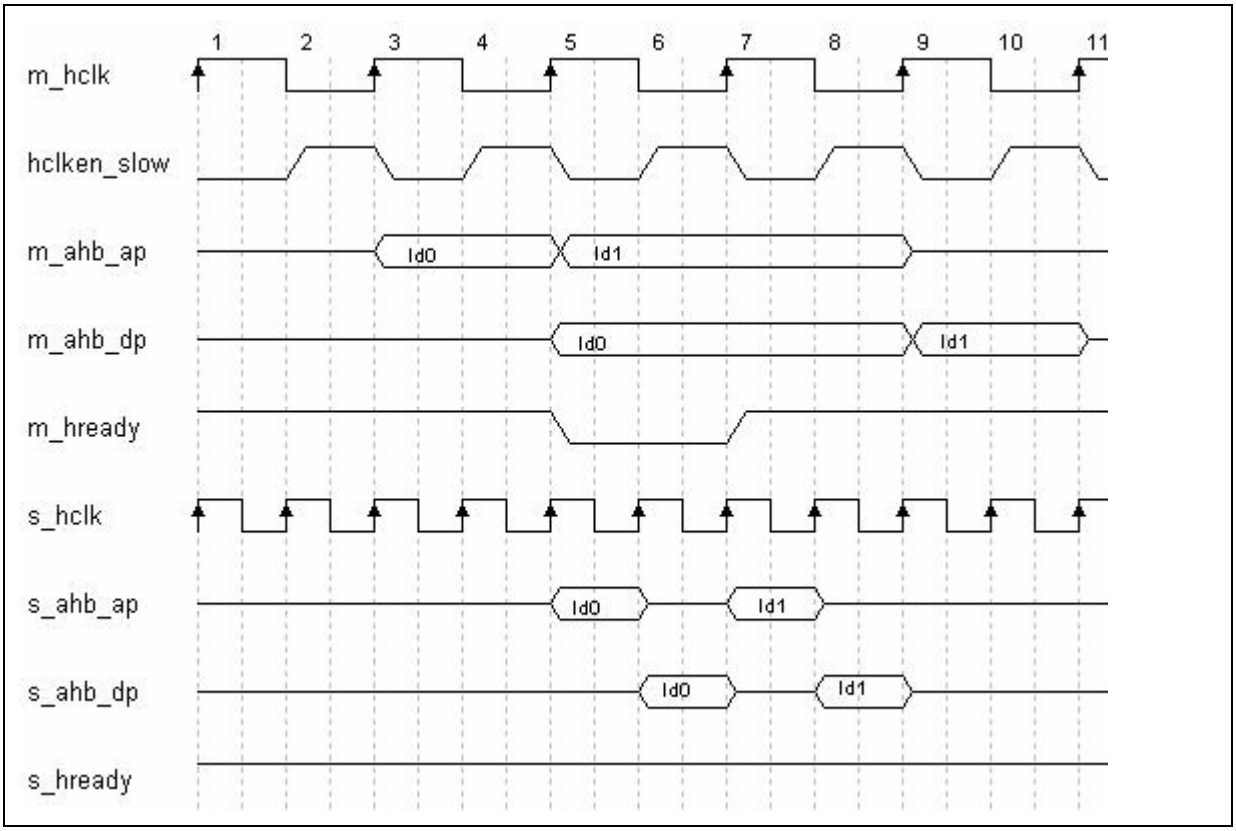


Figure 103. Write, pending read (1:2 timing mode)

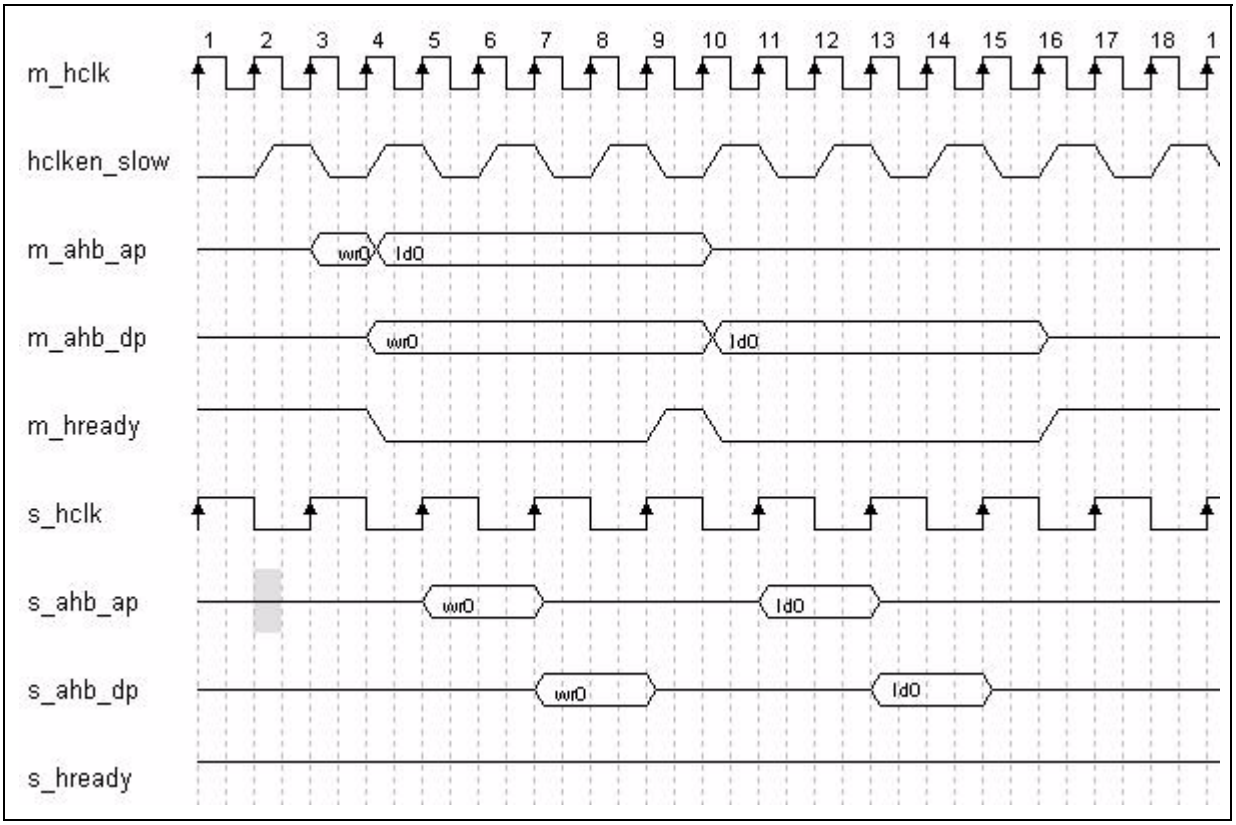


Figure 104. Burst4 read (1:2 timing mode)

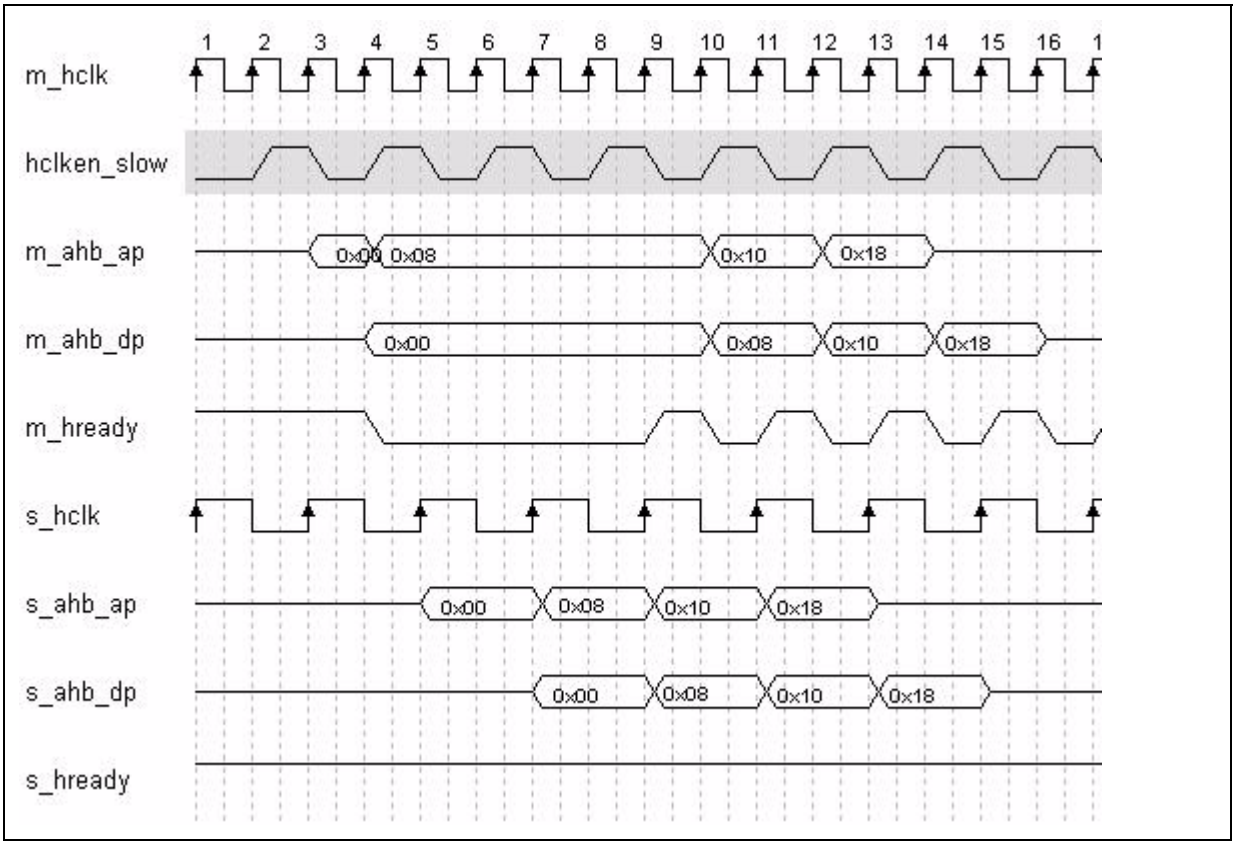
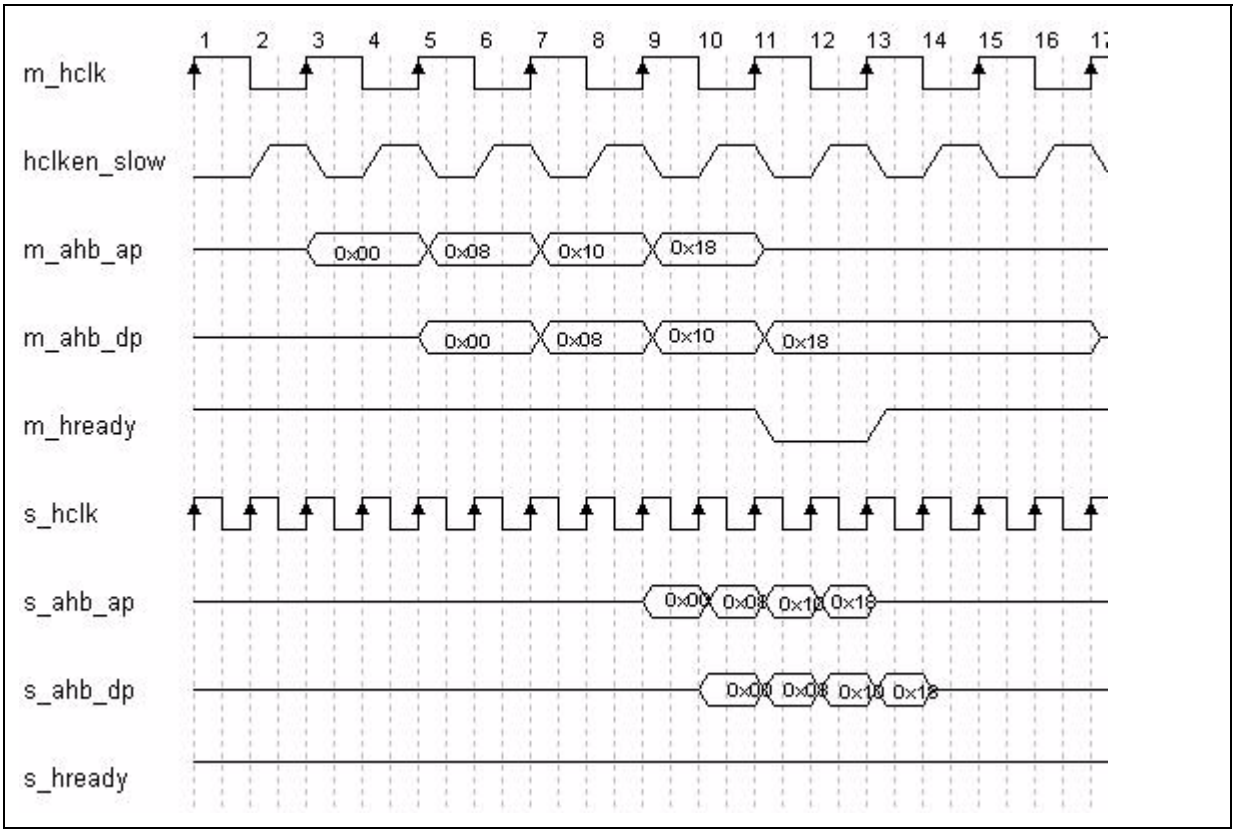


Figure 105. Burst4 write (1:2 timing mode)



17.2.4 64->32 Interface

The 64->32 interface translates a 64-bit access to two 32-bit accesses on the slave side. If there is an hresp on the first 32-bit access to the slave, the iahb_gasket will still do the second 32-bit access. The hresp is aggregated and reported to the master in the data phase of the 64-bit access. One of the 32-bit writes could complete and the other write could return an hresp.

For burst encodings of WRAP4,INCR4 hsize=long, it translates to WRAP8,INCR8 hsize=word.

For burst encodings of WRAP8,INCR8 hsize=long, it translates to WRAP16,INCR16 hsize=word.

Burst encodings WRAP4,8,16 INCR4,8,16 of hsize<long are passed as is through to slave.

All other burst encodings are not optimized and are translated as NSEQ.

Unaligned bursts are not optimized and are translated as NSEQ.

17.2.5 32->64 Interface

The 32->64 interface translates two 32-bit accesses to one 64-bit access on the slave side.

There is no special translation of 32-bit WRAP8,INCR8 to 64-bit WRAP4,INCR4.

Burst encodings WRAP4,8,16 INCR4,8,16 of hsize<long are passed as is through to slave.

All other burst encodings are not optimized and are translated as NSEQ.

Unaligned bursts are translated to NSEQ.

To support initializing ECC in the SRAM where two 32-bit write accesses through the 32->64 gasket needs to get translated to a single 64-bit write access, the master will send aligned (haddr[2]==0) double word transactions with a burst encoding of INCR with length fixed to 2. The iahb_gasket will translate these two 32-bit writes to a single 64-bit NSEQ. It will never report a hresp on the first 32-bit write. A design parameter on the gasket instantiation is used to compare to the master ID and if there is a match, all INCR are treated as an INCR with length fixed to 2.

18 Interrupt Controller (INTC)

18.1 Introduction

The INTC:

- Provides priority-based preemptive scheduling of interrupt requests
- Schedules interrupt requests (IRQs) from software and internal peripherals to one or more processors (PRCs)
- Provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support

This scheduling scheme is suitable for statically scheduled hard real-time systems.

The INTC is targeted to work with a Power Architecture processor and is capable of processing high-demand interrupt sources where the Interrupt Service Routines (ISRs) nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, it is necessary to minimize the interval between the assertion of the interrupt request from the peripheral and the point at which the processor is performing useful work to service the interrupt request. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 64 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

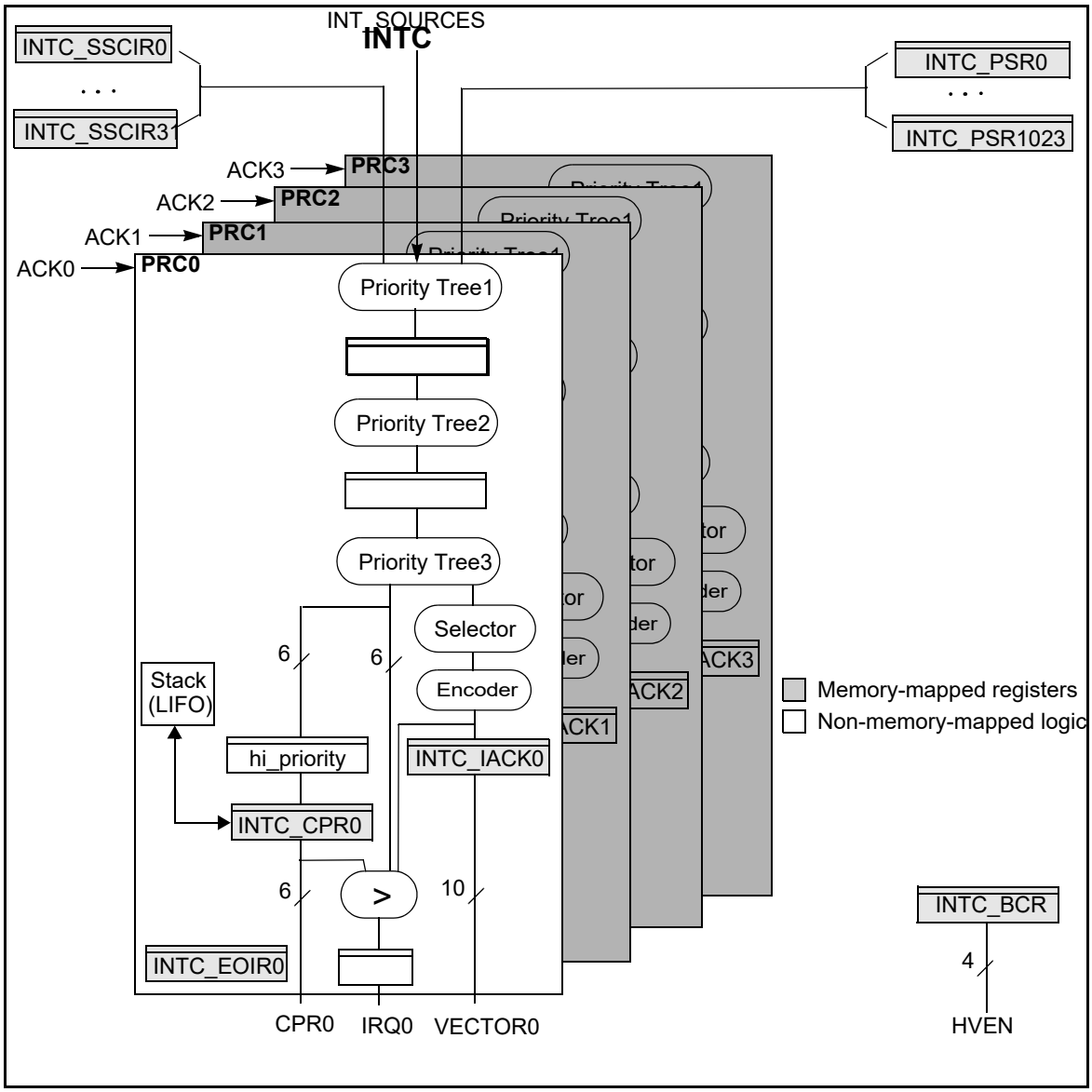
When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the Priority Ceiling Protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource are unable to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

Multiple processors can assert interrupt requests to each other through software-settable interrupt requests. These same 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

Figure 106 shows the block diagram of an INTC with four processors (PRC0...PRC3). The actual number of processors is described in the chip-specific INTC information. The structure of the INTC remains the same for each implemented processor.

Figure 106. Block diagram for an INTC with four processors



18.3 Features

- Each peripheral interrupt source is software-steerable to processor 0, processor 1, processor 2, processor 3, or any combination of interrupt request outputs
- 32 software-settable interrupt request sources
- 10-bit vector
  - Unique vector for each interrupt request source
  - Hardware connection to processor or read from register
- Each interrupt source can be programmed to one of 64 priorities
- Each interrupt source can be triggered by software
- Preemption
  - Preemptive prioritized interrupt requests to processor
  - ISR with higher priority preempts ISRs or tasks with lower priorities
  - Automatic pushing or popping of preempted priority to or from a LIFO
  - Ability to modify the ISR or task priority; modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
  - 3 INTC clock cycles from interrupt request into interrupt request to CPU
- Low latency
  - Two INTC clock cycles from receipt of interrupt request from peripheral to interrupt request to processor; three clock cycles from receipt of software request

18.4 Modes of operation

The interrupt controller has two handshaking modes with the processor: software vector mode and hardware vector mode. The state of the hardware vector enable bit, INTC_BCR[HVEN], determines which mode is used. In Power Architecture devices, software vector mode uses the autovector mode of the Processor for a single entry point for the ISR (16 bytes of vector space available), whereas the hardware vector mode uses the non-autovector mode where the vector offset from the INTC is provided to the Processor (4 bytes for each IRQ).

In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

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 HVEN_PRC*n* bit in the INTC_BCR (see *Section 18.5.2.1, INTC Block Configuration Register (INTC_BCR)*) is negated. The hardware vector enable signal to any processor is driven as negated when its associated HVEN_PRC*n* bit is negated. The vector is read from the INTC_IACKRn registers (see *Section 18.5.2.3, INTC Interrupt Acknowledge Register for Processor n (INTC_IACKRn)*). Reading the INTC_IACKR*n* 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_CPR*n* (see *Section 18.5.2.2, INTC Current Priority Register for Processor n (INTC_CPRn)*) onto

the associated LIFO, and updates PRI in the associated INTC_CPR_PRC*n* with the new priority.

18.4.2 Hardware vector mode

In hardware vector mode, the hardware causes the first instruction that will be executed (when handling the interrupt request to the processor) to be an instruction that is specific to that vector. Therefore the interrupt exception handler is specific to a peripheral or software-settable interrupt request, rather than being common to all of them. The INTC will use hardware vector mode for a given processor when its associated HVEN*n* bit in the INTC_BCR is asserted. The hardware vector enable signal to the associated processor is driven as asserted.

When the interrupt request to the associated processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software-settable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKR*n*, depending on which processor was assigned to handle a given interrupt source.

The assertion of the interrupt acknowledge signal for a given processor pushes the associated PRI value in the associated INTC_CPR*n* register onto the associated LIFO and updates the associated PRI in the associated INTC_CPR*n* register with the new priority. This pushing of the PRI value onto the associated LIFO and updating PRI in the associated INTC_CPR*n* does not occur when the associated interrupt acknowledge signal asserts and the INTC_SSCIRs (see *Section 18.5.2.5, INTC Software Set/Clear Interrupt Registers (INTC_SSCIR0–INTC_SSCIR31)*) is written at a time such that the PRI value in the associated INTC_CPR*n* register would need to be pushed and the previously last pushed PRI value would need to be popped simultaneously. In this case, PRI in the associated INTC_CPR*n* is updated with the new priority, and the associated LIFO is neither pushed or popped.

18.5 Memory map and register definition

18.5.1 Memory map

Table 153 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 153. INTC memory map

Address offset	Register	Location
000h	INTC Block Configuration Register (INTC_BCR)	*on page 311*
008h–00Ch	Reserved	
010h	INTC Current Priority Register for Processor 0 (INTC_CPR0)	*on page 312*
014h	INTC Current Priority Register for Processor 1 (INTC_CPR1)	*on page 312*
018h	INTC Current Priority Register for Processor 2 (INTC_CPR2)	*on page 312*
01Ch	INTC Current Priority Register for Processor 3 (INTC_CPR3)	*on page 312*
020h	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)	*on page 313*

Table 153. INTC memory map(Continued)

Address offset	Register	Location
024h	INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)	*on page 313*
028h	INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)	*on page 313*
02Ch	INTC Interrupt Acknowledge Register for Processor 3 (INTC_IACKR3)	*on page 313*
030h	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)	*on page 314*
034h	INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)	*on page 314*
038h	INTC End Of Interrupt Register for Processor 2 (INTC_EOIR2)	*on page 314*
03Ch	INTC End Of Interrupt Register for Processor 3 (INTC_EOIR3)	*on page 314*
040h–05Fh	INTC Software Set/Clear Interrupt Register 0 (INTC_SSCIR0)– INTC Software Set/Clear Interrupt Register 31(INTC_SSCIR31)	*on page 314*
060h–85Eh	INTC Priority Select Registers (INTC_PSR0 – INTC_PSR1023)	*on page 315*
860h-FFC	Reserved	

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_SSCIR*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_IACKR*n*) are the same regardless of the size of the read. In either software or hardware vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR*n*) 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: 000h 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	HVEN3	0	0	0	HVEN2	0	0	0	HVEN1	0	0	0	HVEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 107. INTC Block Configuration Register (INTC_BCR)

Table 154. INTC_BCR field descriptions

Field	Description
HVEN*n*	Hardware vector enable. Controls whether the INTC is in hardware vector mode or software vector mode. See *Section 18.4, Modes of operation*, for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

18.5.2.2 INTC Current Priority Register for Processor *n* (INTC_CPR*n*)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in *Section 18.4.2, Hardware vector mode*.

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

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, Ensuring coherency, for example code to ensure coherency.

Offsets: 010h (INTC_CPR0), 014h (INTC_CPR1), 018h (INTC_CPR2), 01Ch (INTC_CPR3)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	PRI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 108. INTC Current Priority Register for Processor *n* (INTC_CPR*n*)

Table 155. INTC_CPR*n* field descriptions

Field	Description
PRI	Priority of the currently executing ISR, according to the field values 63 (highest priority) down to 0 (lowest priority).

18.5.2.3 INTC Interrupt Acknowledge Register for Processor *n* (INTC_IACKR*n*)

The Interrupt Acknowledge Register for Processor *n* 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_IACKR*n* has side effects from reads. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKR*n* does not have side effects in hardware vector mode.

Offsets: 020h (INTC_IACKR0), 024h (INTC_IACKR1), 028h (INTC_IACKR2), 02Ch (INTC_IACKR3)

Access: User write always; non-speculative read(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VTBA[19:4]															
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	VTBA[3:0]				INTVEC										0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 109. INTC Interrupt Acknowledge Register for Processor *n* (INTC_IACKR*n*)

1. When the corresponding INTC_BCR[HVEN*n*] = 1, a read of the INTC_IACKR has no side effects.

Table 156. INTC_IACKRn field descriptions

Field	Description
VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.

18.5.2.4 INTC End of Interrupt Register for Processor *n* (INTC_EOIR*n*)

Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

Offsets: 030h (INTC_EOIR0)
034h (INTC_EOIR1)
038h (INTC_EOIR2)
03Ch (INTC_EOIR3)

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	EOI[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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	EOI[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 110. INTC End of Interrupt Register for Processor *n* (INTC_EOIR*n*)

Table 157. INTC_EOIR*n* field descriptions

Field	Description
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 Registers (INTC_SSCIR0–INTC_SSCIR31)

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 111 shows the structure of the first INTC_SSCIR. The other registers follow the same structure.

Offset: 040h 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 111. INTC Software Set/Clear Interrupt Registers (INTC_SSCIR*n*)

Table 158. INTC_SSCIR*n* field descriptions

Field	Description
SET	Set flag bits. Writing a 1 will set the corresponding CLR bit. Writing a 0 will have no effect. Each SET is read as a 0.
CLR	Clear flag bits. 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 Registers (INTC_PSR0 – INTC_PSR1023)

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 112 and *Figure 113* 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.

Note: Interrupts are not detected while the PSRs are being written.

Offset: 060h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRC_SEL0				0	0	0	0	0	0	PRI0					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 112. INTC Priority Select Register 0 (INTC_PSR0)

Offset: 85Eh Access: User read/write

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PRC_SEL1023				0	0	0	SWT1023	0	0	PRI1023					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 113. INTC Priority Select Register 1023 (INTC_PSR1023)

Table 159. INTC_PSR*n* field descriptions

Field	Description
PRC_SEL*n*	Processor select bits. See *Table 160*. If an interrupt source is enabled, PRC_SEL*n*[3:0] selects whether the interrupt request is to be sent to processor 0, processor 1, processor 2, processor 3, or any combination.
SWT*n*	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. The SWT function works both in hardware and software vector mode. It is only cleared by the enabled Processor(s) defined in the PRC_SEL*n* field of the PSR.
PRI*n*	Priority select. Selects the priority for the interrupt requests. PRI has values 63 (highest priority) down to 0 (lowest priority).

Table 160. Selected Processor for Interrupt Request

PRC_SEL*n*[3:0]	Meaning
0000	No interrupt request sent
0001	Interrupt request sent to processors 3
0010	Interrupt request sent to processors 2
0011	Interrupt request sent to processors 2, 3
0100	Interrupt request sent to processors 1
0101	Interrupt request sent to processors 1, 3
0110	Interrupt request sent to processors 1, 2
0111	Interrupt request sent to processors 1, 2, 3
1000	Interrupt request sent to processors 0
1001	Interrupt request sent to processors 0, 3
1010	Interrupt request sent to processors 0, 2
1011	Interrupt request sent to processors 0, 2, 3
1100	Interrupt request sent to processors 0, 1
1101	Interrupt request sent to processors 0, 1, 3
1110	Interrupt request sent to processors 0, 1, 2
1111	Interrupt request sent to processors 0, 1, 2, 3

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_CPR*n*, 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_CPR*n* 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_SELn] 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_CPR*n*. 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 106* 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_IACKR*n*, and if in hardware vector mode, for the interrupt vector provided to the processor.

18.6.2.1.1 Priority tree

The priority tree for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software-settable. The output of the priority tree is the highest of those priorities assigned to a given 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_CPR*n*. If the comparator detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the associated processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in the associated INTC_CPR*n*, or the PRI value in the associated INTC_CPR*n* is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in the associated INTC_CPR*n* 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_CPR*n*.

Another function of the comparator is to signal an update of the INTC_IACKR*n* 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_CPR*n* 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_CPR*n*. 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_CPR*n* does not need to be loaded from the associated INTC_CPR*n* 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_CPR*n*.

The PRI value in the associated INTC_CPR*n* is pushed onto the LIFO when the associated INTC_IACKR*n* is read in software vector mode or the interrupt acknowledge signal from the associated processor is asserted in hardware vector mode. The priority is popped into PRI in the associated INTC_CPR*n* whenever the associated INTC_EOIR*n* is written. An exception case in hardware vector mode to this behavior is described in *Section 18.4.2, Hardware vector mode*.

Although the INTC supports up to 64 priorities, an ISR executing with PRI in the INTC_CPR*n* equal to 15 will not be preempted. Therefore, the LIFO supports the stacking of 15 priorities. However, the LIFO is only 14 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 15 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 114*.

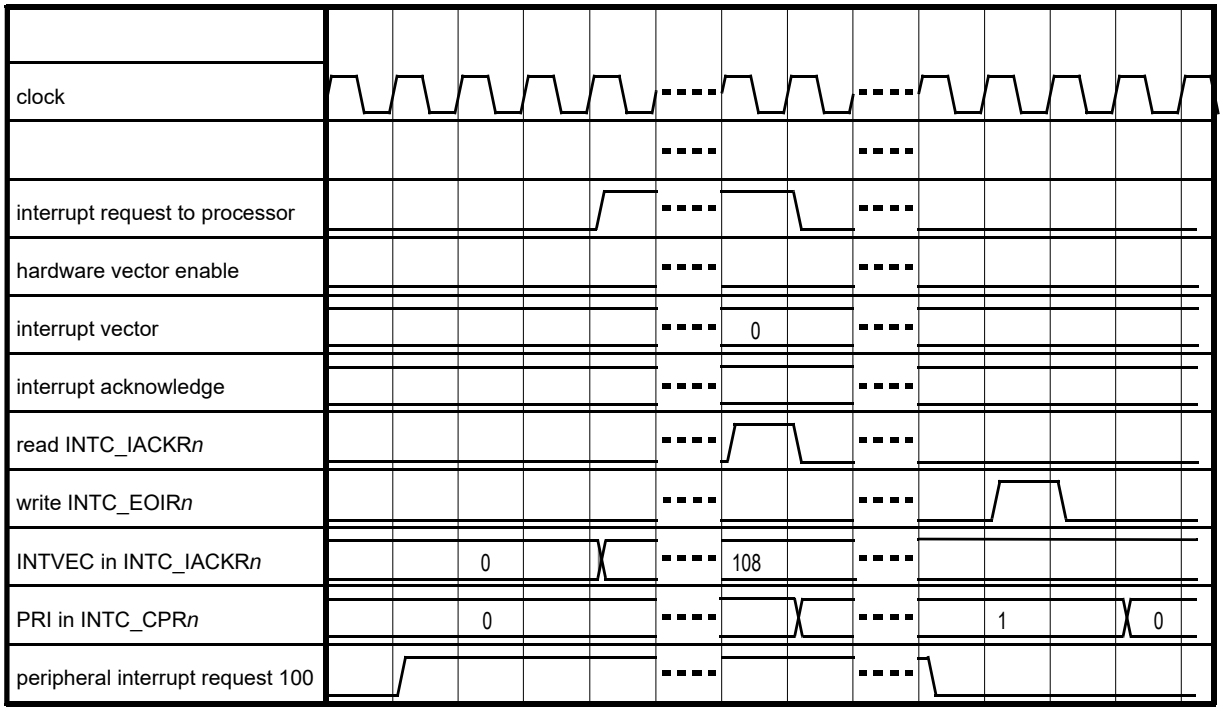
A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in *Figure 114*. The INTC examines the peripheral and software-settable interrupt requests. When it finds an asserted peripheral or software-settable interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* 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, Software vector mode*.

18.6.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR*n* 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_CPR*n*. 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_CPR*n* 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 114. Timing diagram of software vector mode handshaking

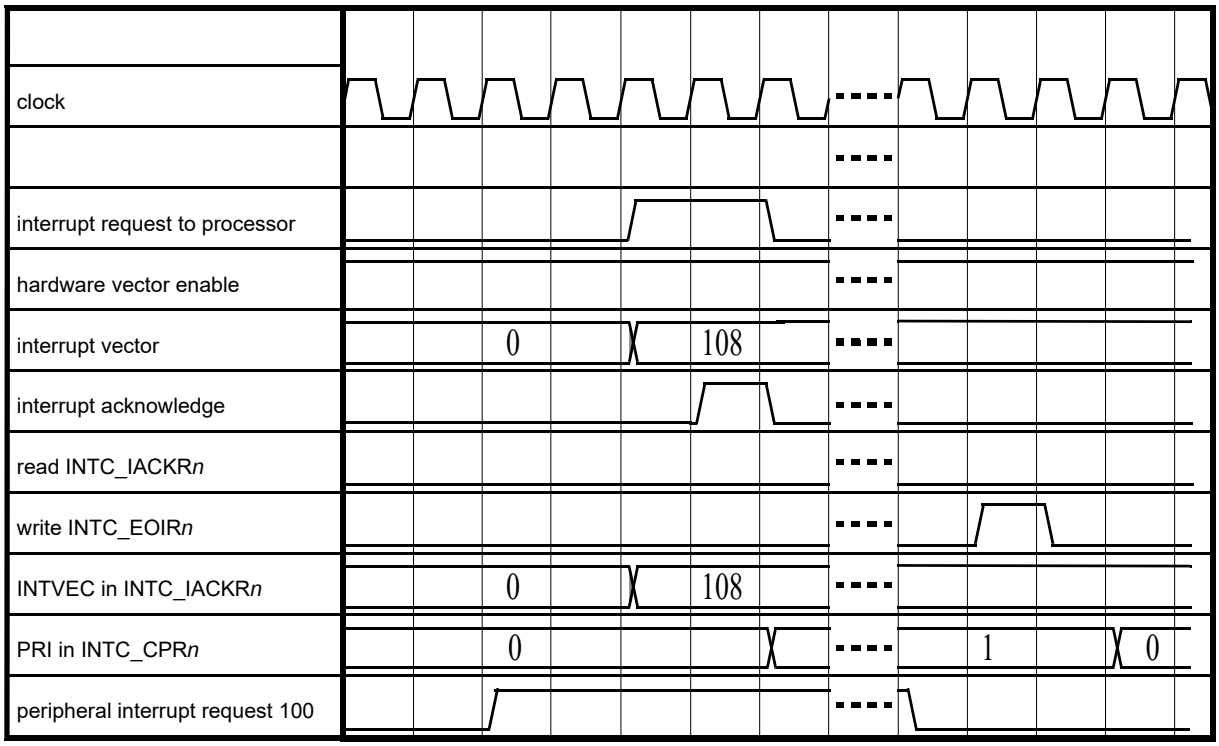


18.6.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in *Figure 115*. As in software vector mode, the INTC examines the peripheral and software-settable interrupt requests, and when it finds an asserted interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* is updated with the preempting peripheral or software-settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the associated processor is asserted. In addition, the value of the interrupt vector to the associated processor matches the value of the INTVEC field in the associated INTC_IACKR*n*. The rest of the handshaking is described in *Section 18.4.2, Hardware vector mode*.

The handshaking near the end of the interrupt exception handler, that is the writing to the associated INTC_EOIR*n*, is the same as in software vector mode. See *Section 18.6.3.1.2, End of interrupt exception handler*.

Figure 115. Timing diagram for hardware 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_CPRn registers is 63. These reset values will prevent the INTC from asserting the interrupt request to the processors. 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 HVENn in INTC_BCR.
  - Configure VTBA*n* in INTC_IACKR*n.*
  - 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_CPR*n* to zero.
  - Enable processor(s) 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)


lisr3,INTC_IACKRn@ha# form adjusted upper half of INTC_IACKRn address
lwzr3,INTC_IACKRn@l(r3)# load INTC_IACKRn, which clears request to
processor
lwzr3,0x0(r3)# load address of ISR from vector table
wrteei1    # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (Embedded
Binary Application Interface) (not shown)

mtlrr3     # move the INTC_IACKRn 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
lisr3,INTC_EOIRn@ha# form adjusted upper half of INTC_EOIRn address
li r4,0x0  # form 0 to write to INTC_EOIRn
wrteei0    # disable processor recognition of interrupts
stwr4,INTC_EOIRn@l(r3) # store to INTC_EOIRn, 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
```

18.7.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector.

```
interrupt_exception_handlern:
b  interrupt_exception_handler_continuedn# 16 bytes available, branch to
continue


interrupt_exception_handler_continuedn:
code to create stack frame, save working register, and save SRR0 and SRR1
(not shown)

wrteei1    # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (not shown)

bl ISRn    # branch to ISR for interrupt with vector n


epilog:
code to restore most of context required by Power Architecture EABI (not
shown)
```

```
# Popping the LIFO after the restoration of most of the context and the
disabling of processor
# recognition of interrupts eases the calculation of the maximum stack depth
at the cost of
# postponing the servicing of the next interrupt request.
mbar       # ensure store to clear flag bit has completed
lisr3,INTC_EOIRn@ha# form adjusted upper half of INTC_EOIRn address
li r4,0x0  # form 0 to write to INTC_EOIRn
wrteei0    # disable processor recognition of interrupts
stwr4,INTC_EOIRn@l(r3)# store to INTC_EOIRn, informing INTC to lower
priority

SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi



ISRn:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr        # branch to epilog
```

18.7.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC_CPR*n* 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_CPR*n* priority 0 and outside the control of the RTOS, the RTOS executes at INTC_CPR*n* priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPR*n* 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_CPR*n* 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 selected 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 161* shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 161. Order of ISR execution example

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 (1)	ISR208	ISR308	ISR408	interrupt exception handler	
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_EOIR*n*.						X	1
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR*n*.						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_CPR*n*) 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_CPR*n* to 3, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR*n* 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_CPR*n*. 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_CPR*n* can be made by those system services with the code sequence:

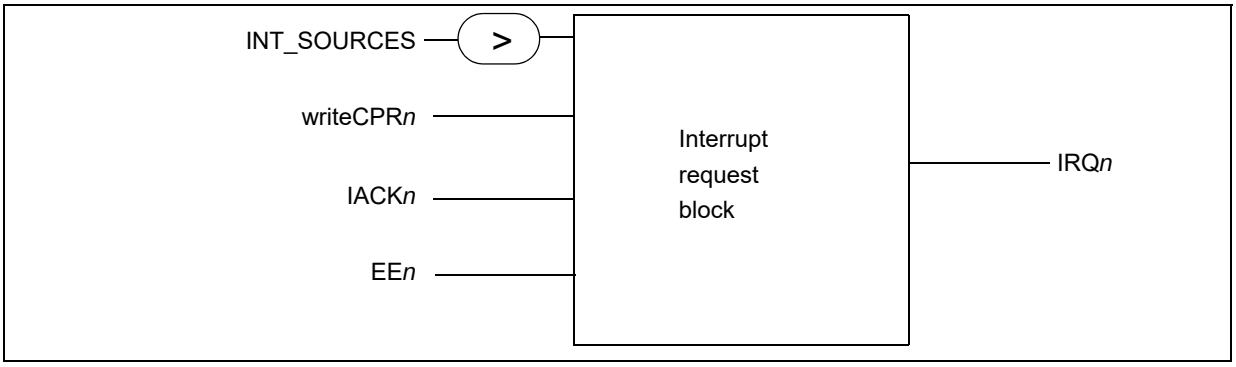
```
GetResource:
   wrteei 0# disable external interrupts to the Processor
   raise PRI# Write to CPR, cache inhibit, guarded
   mbar # flush out writes from store buffer
   wrteei 1# enable external interrupts to the Processor
   isync# re-fetch Processor pipeline

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 106*, the interrupt request logic to the processor is shown in *Figure 116*.

Figure 116. 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_CPR*n*, then interrupt request to the processor (IRQ*n*) is asserted. It will stay asserted until one of the following conditions is true:

- Interrupt acknowledge (IACK) is asserted
- If the EE*n* bit has been cleared by the wrteei instruction (see *Figure 116*), IRQ*n* will be re-evaluated when processor *n* writes to the INTC_CPR*n* 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_CPR*n*.

18.7.5.2.2 Raised priority preserved

Before the instruction after the GetResource system service executes, all pending transactions have completed. 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. The shared coherent data block now can be accessed coherently. *Figure 117* shows the timing diagram for this scenario, and *Table 162* explains the events. The example is for software vector mode. Except for the method of retrieving the vector and acknowledging the interrupt request to the processor, hardware vector mode is identical.

Figure 117. Timing diagram of raised priority preserved

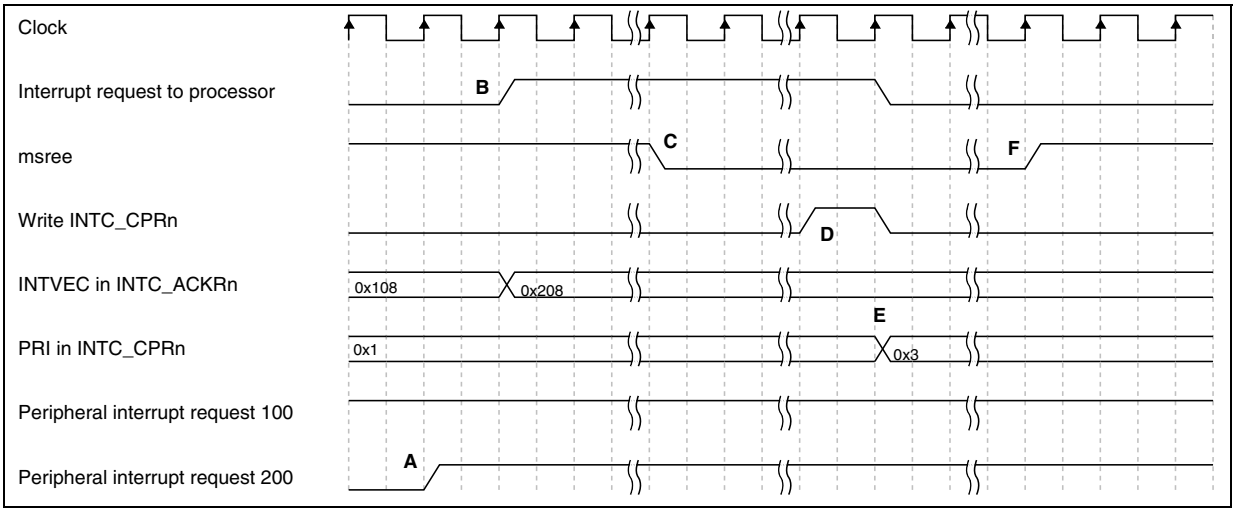


Table 162. Raised priority preserved events

Event	Description
A	Peripheral interrupt request 200 of priority 2 asserts during execution of ISR108 running at priority 1.
B	Interrupt request to processor asserts. INTVEC in INTC_IACKRn updates with vector for that peripheral interrupt request.
C	Write to msree bit to disable external interrupts
D	ISR108 writes to INTC_CPRn to raise priority to 3 before accessing shared coherent data block.
E	PRI in INTC_CPRn now at 3, reflecting the write. This write, just before accessing data block, is the last instruction the processor executes before being interrupted.
F	Write to msree bit to enable external interrupts

18.7.6 Selecting priorities according to the request rates and deadlines

The selection of the priorities for the ISRs can be made using Rate Monotonic Scheduling (RMS) or a superset of it, Deadline Monotonic Scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC supports 64 priorities, which may be many fewer than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, and so on. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion, which reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

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 and for processors to interrupt other processors in a multiple processor system.

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_CPR*n* with the interrupt acknowledge. The ISR, however, can have a portion of it which does not need to be executed at this higher priority. Therefore, executing this later portion which does not need to 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.

18.7.7.2 Scheduling an ISR on another processor

Since the SET bits in the INTC_SSCIR*n* are memory mapped, processors in multiple processor systems can schedule ISRs on the other processors. One possible application is if one processor simply wants to command another processor to perform a piece of work, and the initiating processor does not need to use the results of that work. If the initiating processor needs to know if the processor executing the software-settable ISR has not completed the work before asking it to again execute that ISR, it can check if the corresponding CLR bit in INTC_SSCIR*n* is asserted before again writing a 1 to the SET bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. The procedure is that the first processor writes a 1 to a SET bit on the second processor. The second processor, after accessing the block of data, clears the corresponding CLR bit and then writes 1 to a SET bit on the first processor, informing it that it now can access the block of data.

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, Scheduling a lower priority portion of an ISR*) 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_CPRn 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, End of interrupt exception handler* 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

There are multiple methods for tracking interrupts in a system, one of which is described here using existing hardware (LIFO) within the INTC. 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_CPR*n*). To avoid a lower-level interrupt being serviced because the popping of the LIFO updates the INTC_CPR*n*, the processor recognition of interrupts should be disabled when examining LIFO contents. The pseudo-code is as follows:

```
wrteei 0 # disable processor recognition of external interrupts
```

```
   oldCPRn = INTC_CPRn; # save INTC_CPRn
   (branch to the pop_lifo)

pop_lifo:
   store INTC_EIORn # pop INTC_CPR from LIFO, examine PRI, etc...
   examine INTC_CPRn[PRI], and store onto stack
   if PRI is not zero or value when interrupts were enabled, branch to
pop_lifo
   branch to push_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_CPRn
   load INTC_IACKn # IACK; push INTC_CPRn into LIFO
   if stacked PRI values are not depleted, branch to push_lifo
   INTC_CPRn = oldCPRn; # restore original INTC_CPRn and re-evaluate
pending
            external interrupts
   wrteei 1 # enable processor recognition of interrupts
```

Note: Disabling the processor recognition of interrupts during LIFO examination will introduce latency, but none of the interrupt requests will be missed.

18.8 Interrupt sources

The list of interrupt sources is chip-specific. For this list, see the chip-specific INTC information.

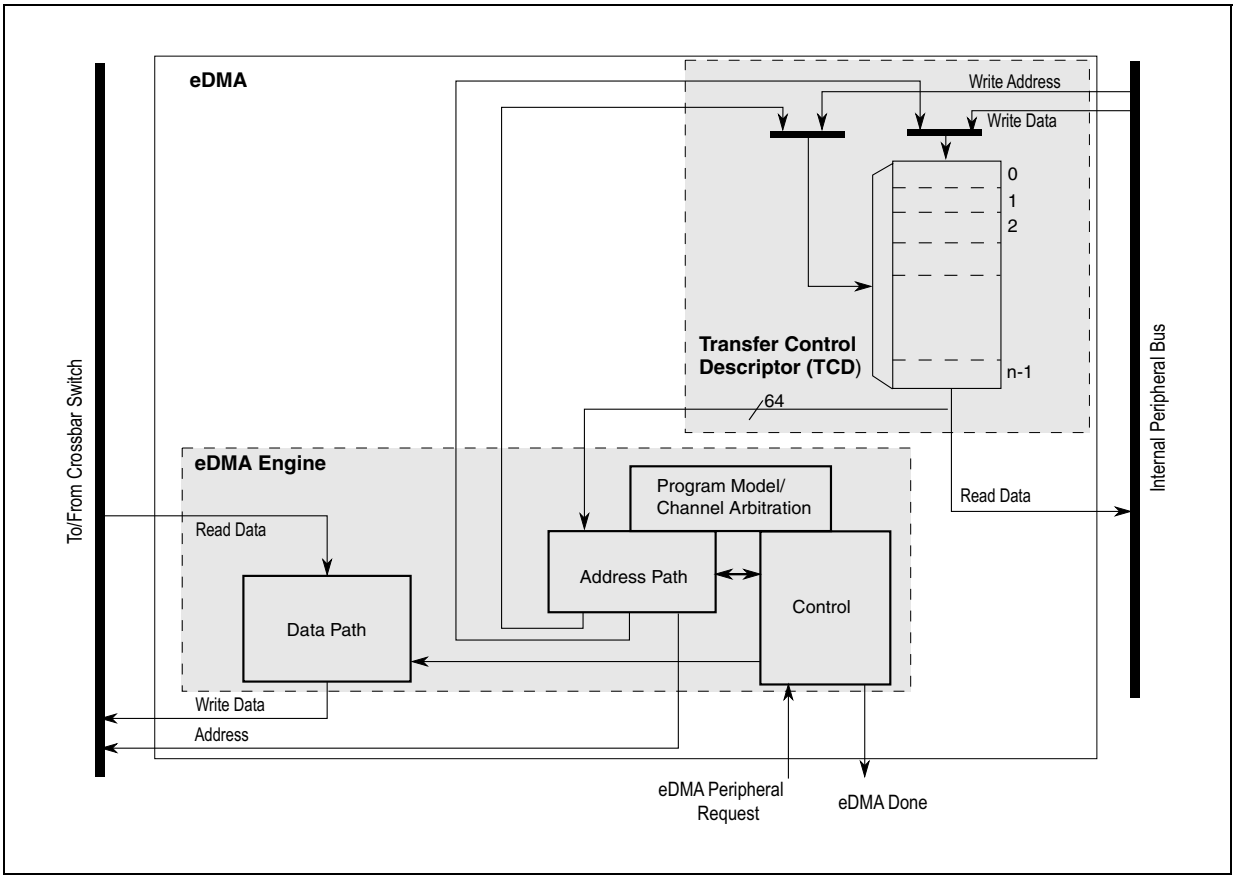
19 Enhanced Direct Memory Access (eDMA)

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 118 illustrates the eDMA module.

Figure 118. 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- and 32-byte burst transfers
  - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests (one per channel)
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via optional interrupt requests
  - One interrupt per channel, optionally asserted at completion of major iteration count
  - Optional error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Optional support for scatter/gather DMA processing
- Support for complex data structures
- Support to cancel transfers via software

Throughout this chapter, *n* is used to reference the channel number.

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.

Some registers are implemented as two 32-bit registers, and include an H and L suffix, signaling the high and low portions of the control function. *Table 163* 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 11 registers of 16 or 32 bits.

Reading reserved bits in a register returns the value of zero and writes to reserved bits in a register are ignored. Reading or writing a reserved memory location generates a bus error.

Table 163. DMA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_0000	Control Register (DMA_CR)	32	R/W	0000_0000h	*on page 347*
FC0A_0004	Error Status Register (DMA_ES)	32	R	0000_0000h	*on page 349*
FC0A_0008–FC0A_000B	Reserved				
FC0A_000C	Enable Request Register Low (DMA_ERQL)	32	R/W	0000_0000h	*on page 350*
FC0A_0010–FC0A_0013	Reserved				
FC0A_0014	Enable Error Interrupt Register Low (DMA_EEIL)	32	R/W	0000_0000h	*on page 352*
FC0A_0018	Set Enable Request Register (DMA_SERQ)	8	W (always reads 0)	00h	*on page 354*
FC0A_0019	Clear Enable Request Register (DMA_CERQ)	8	W (always reads 0)	00h	*on page 355*
FC0A_001A	Set Enable Error Interrupt Register (DMA_SEEI)	8	W (always reads 0)	00h	*on page 355*
FC0A_001B	Clear Enable Error Interrupt Register (DMA_CEEI)	8	W (always reads 0)	00h	*on page 356*
FC0A_001C	Clear Interrupt Request Register (DMA_CINT)	8	W (always reads 0)	00h	*on page 357*

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_001D	Clear Error Register (DMA_CERR)	8	W (always reads 0)	00h	on page 357
FC0A_001E	Set START Bit Register (DMA_SSRT)	8	W (always reads 0)	00h	on page 358
FC0A_001F	Clear DONE Status Bit Register (DMA_CDNE)	8	W (always reads 0)	00h	on page 358
FC0A_0020–FC0A_0023	Reserved				
FC0A_0024	Interrupt Request Register Low (DMA_INTL)	32	R/W	0000_0000h	on page 359
FC0A_0028–FC0A_002B	Reserved				
FC0A_002C	Error Register Low (DMA_ERRL)	32	R/W	0000_0000h	on page 361
FC0A_0030–FC0A_0033	Reserved				
FC0A_0034	Hardware Request Status Register Low (DMA_HRSL)	32	R/W	0000_0000h	on page 363
FC0A_0038–FC0A_00FF	Reserved				
FC0A_0100	Channel 0 Priority Register (DMA_DCHPRI0)	8	R/W	0Xh	on page 365
FC0A_0101	Channel 1 Priority Register (DMA_DCHPRI1)	8	R/W	0Xh	on page 365
FC0A_0102	Channel 2 Priority Register (DMA_DCHPRI2)	8	R/W	0Xh	on page 365
FC0A_0103	Channel 3 Priority Register (DMA_DCHPRI3)	8	R/W	0Xh	on page 365
FC0A_0104	Channel 4 Priority Register (DMA_DCHPRI4)	8	R/W	0Xh	on page 365
FC0A_0105	Channel 5 Priority Register (DMA_DCHPRI5)	8	R/W	0Xh	on page 365
FC0A_0106	Channel 6 Priority Register (DMA_DCHPRI6)	8	R/W	0Xh	on page 365
FC0A_0107	Channel 7 Priority Register (DMA_DCHPRI7)	8	R/W	0Xh	on page 365
FC0A_0108	Channel 8 Priority Register (DMA_DCHPRI8)	8	R/W	0Xh	on page 365
FC0A_0109	Channel 9 Priority Register (DMA_DCHPRI9)	8	R/W	0Xh	on page 365

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_010A	Channel 10 Priority Register (DMA_DCHPRI10)	8	R/W	0Xh	on page 365
FC0A_010B	Channel 11 Priority Register (DMA_DCHPRI11)	8	R/W	0Xh	on page 365
FC0A_010C	Channel 12 Priority Register (DMA_DCHPRI12)	8	R/W	0Xh	on page 365
FC0A_010D	Channel 13 Priority Register (DMA_DCHPRI13)	8	R/W	0Xh	on page 365
FC0A_010E	Channel 14 Priority Register (DMA_DCHPRI14)	8	R/W	0Xh	on page 365
FC0A_010F	Channel 15 Priority Register (DMA_DCHPRI15)	8	R/W	0Xh	on page 365
FC0A_0110–FC0A_013F	Reserved				
FC0A_0140	Channel 0 Master ID Register (DMA_DCHMID0)	8	R/W	00h	on page 365
FC0A_0141	Channel 1 Master ID Register (DMA_DCHMID1)	8	R/W	00h	on page 365
FC0A_0142	Channel 2 Master ID Register (DMA_DCHMID2)	8	R/W	00h	on page 365
FC0A_0143	Channel 3 Master ID Register (DMA_DCHMID3)	8	R/W	00h	on page 365
FC0A_0144	Channel 4 Master ID Register (DMA_DCHMID4)	8	R/W	00h	on page 365
FC0A_0145	Channel 5 Master ID Register (DMA_DCHMID5)	8	R/W	00h	on page 365
FC0A_0146	Channel 6 Master ID Register (DMA_DCHMID6)	8	R/W	00h	on page 365
FC0A_0147	Channel 7 Master ID Register (DMA_DCHMID7)	8	R/W	00h	on page 365
FC0A_0148	Channel 8 Master ID Register (DMA_DCHMID8)	8	R/W	00h	on page 365
FC0A_0149	Channel 9 Master ID Register (DMA_DCHMID9)	8	R/W	00h	on page 365
FC0A_014A	Channel 10 Master ID Register (DMA_DCHMID10)	8	R/W	00h	on page 365
FC0A_014B	Channel 11 Master ID Register (DMA_DCHMID11)	8	R/W	00h	on page 365
FC0A_014C	Channel 12 Master ID Register (DMA_DCHMID12)	8	R/W	00h	on page 365

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_014D	Channel 13 Master ID Register (DMA_DCHMID13)	8	R/W	00h	on page 365
FC0A_014E	Channel 14 Master ID Register (DMA_DCHMID14)	8	R/W	00h	on page 365
FC0A_014F	Channel 15 Master ID Register (DMA_DCHMID15)	8	R/W	00h	on page 365
FC0A_0150–FC0A_0FFF	Reserved				
FC0A_1000	TCD0 Source Address (DMA_TCD0_SADDR)	32	R/W	0000_000Xh	on page 366
FC0A_1004	TCD0 Transfer Attributes (DMA_TCD0_ATTR)	16	R/W	000Xh	on page 367
FC0A_1006	TCD0 Signed Source Address Offset (DMA_TCD0_SOFF)	16	R/W	000Xh	on page 368
FC0A_1008	TCD0 Minor Byte Count (Minor Loop Disabled) (DMA_TCD0_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD0_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD0_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_100C	(DMA_TCD0_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_1010	(DMA_TCD0_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_1014	(DMA_TCD0_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD0 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD0_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_1016	(DMA_TCD0_DOFF)	16	R/W	000Xh	on page 374
FC0A_1018	(DMA_TCD0_DLASTSGA)	32	R/W	0000_000Xh	on page 375
FC0A_101C	(DMA_TCD0_BITER_ELINKYES)	16	R/W	000Xh	on page 375
	(DMA_TCD0_BITER_ELINKNO)	16	R/W	000Xh	on page 376
FC0A_101E	(DMA_TCD0_CSR)	16	R/W	000Xh	on page 377
FC0A_1020	TCD1 Source Address (DMA_TCD1_SADDR)	32	R/W	0000_000Xh	on page 366
FC0A_1024	TCD1 Transfer Attributes (DMA_TCD1_ATTR)	16	R/W	000Xh	on page 367
FC0A_1026	TCD1 Signed Source Address Offset (DMA_TCD1_SOFF)	16	R/W	000Xh	on page 368

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1028	TCD1 Minor Byte Count (Minor Loop Disabled) (DMA_TCD1_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD1_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD1_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_102C	(DMA_TCD1_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_1030	(DMA_TCD1_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_1034	(DMA_TCD1_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD1 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD1_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_1036	(DMA_TCD1_DOFF)	16	R/W	000Xh	on page 374
FC0A_1038	(DMA_TCD1_DLASTSGA)	32	R/W	0000_000Xh	on page 375
FC0A_103C	(DMA_TCD1_BITER_ELINKYES)	16	R/W	000Xh	on page 375
	(DMA_TCD1_BITER_ELINKNO)	16	R/W	000Xh	on page 376
FC0A_103E	(DMA_TCD1_CSR)	16	R/W	000Xh	on page 377
FC0A_1040	TCD2 Source Address (DMA_TCD2_SADDR)	32	R/W	0000_000Xh	on page 366
FC0A_1044	TCD2 Transfer Attributes (DMA_TCD2_ATTR)	16	R/W	000Xh	on page 367
FC0A_1046	TCD2 Signed Source Address Offset (DMA_TCD2_SOFF)	16	R/W	000Xh	on page 368
FC0A_1048	TCD2 Minor Byte Count (Minor Loop Disabled) (DMA_TCD2_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD2_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD2_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_104C	(DMA_TCD2_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_1050	(DMA_TCD2_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_1054	(DMA_TCD2_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD2 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD2_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_1056	(DMA_TCD2_DOFF)	16	R/W	000Xh	on page 374
FC0A_1058	(DMA_TCD2_DLASTSGA)	32	R/W	0000_000Xh	on page 375
FC0A_105C	(DMA_TCD2_BITER_ELINKYES)	16	R/W	000Xh	on page 375
	(DMA_TCD2_BITER_ELINKNO)	16	R/W	000Xh	on page 376

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_105E	(DMA_TCD2_CSR)	16	R/W	000Xh	on page 377
FC0A_1060	TCD3 Source Address (DMA_TCD3_SADDR)	32	R/W	0000_000Xh	on page 366
FC0A_1064	TCD3 Transfer Attributes (DMA_TCD3_ATTR)	16	R/W	000Xh	on page 367
FC0A_1066	TCD3 Signed Source Address Offset (DMA_TCD3_SOFF)	16	R/W	000Xh	on page 368
FC0A_1068	(DMA_TCD3_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD3_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD3_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_106C	(DMA_TCD3_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_1070	(DMA_TCD3_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_1074	(DMA_TCD3_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD3 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD3_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_1076	(DMA_TCD3_DOFF)	16	R/W	000Xh	on page 374
FC0A_1078	(DMA_TCD3_DLASTSGA)	32	R/W	0000_000Xh	on page 375
FC0A_107C	(DMA_TCD3_BITER_ELINKYES)	16	R/W	000Xh	on page 375
	(DMA_TCD3_BITER_ELINKNO)	16	R/W	000Xh	on page 376
FC0A_107E	(DMA_TCD3_CSR)	16	R/W	000Xh	on page 377
FC0A_1080	TCD4 Source Address (DMA_TCD4_SADDR)	32	R/W	0000_000Xh	on page 366
FC0A_1084	TCD4 Transfer Attributes (DMA_TCD4_ATTR)	16	R/W	000Xh	on page 367
FC0A_1086	TCD4 Signed Source Address Offset (DMA_TCD4_SOFF)	16	R/W	000Xh	on page 368
FC0A_1088	(DMA_TCD4_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD4_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD4_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_108C	(DMA_TCD4_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_1090	(DMA_TCD4_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_1094	(DMA_TCD4_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD4 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD4_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_1096	(DMA_TCD4_DOFF)	16	R/W	000Xh	on page 374

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1098	(DMA_TCD4_DLASTSGA)	32	R/W	0000_000Xh	*on page 375*
FC0A_109C	(DMA_TCD4_BITER_ELINKYES)	16	R/W	000Xh	*on page 375*
	(DMA_TCD4_BITER_ELINKNO)	16	R/W	000Xh	*on page 376*
FC0A_109E	(DMA_TCD4_CSR)	16	R/W	000Xh	*on page 377*
FC0A_10A0	TCD5 Source Address (DMA_TCD5_SADDR)	32	R/W	0000_000Xh	*on page 366*
FC0A_10A4	TCD5 Transfer Attributes (DMA_TCD5_ATTR)	16	R/W	000Xh	*on page 367*
FC0A_10A6	TCD5 Signed Source Address Offset (DMA_TCD5_SOFF)	16	R/W	000Xh	*on page 368*
FC0A_10A8	(DMA_TCD5_NBYTES_MLNO)	32	R/W	0000_000Xh	*on page 369*
	(DMA_TCD5_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	*on page 369*
	(DMA_TCD5_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	*on page 370*
FC0A_10AC	(DMA_TCD5_SLAST)	32	R/W	0000_000Xh	*on page 371*
FC0A_10B0	(DMA_TCD5_DADDR)	32	R/W	0000_000Xh	*on page 372*
FC0A_10B4	(DMA_TCD5_CITER_ELINKYES)	16	R/W	000Xh	*on page 372*
	TCD5 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD5_CITER_ELINKNO)	16	R/W	000Xh	*on page 373*
FC0A_10B6	(DMA_TCD5_DOFF)	16	R/W	000Xh	*on page 374*
FC0A_10B8	(DMA_TCD5_DLASTSGA)	32	R/W	0000_000Xh	*on page 375*
FC0A_10BC	(DMA_TCD5_BITER_ELINKYES)	16	R/W	000Xh	*on page 375*
	(DMA_TCD5_BITER_ELINKNO)	16	R/W	000Xh	*on page 376*
FC0A_10BE	(DMA_TCD5_CSR)	16	R/W	000Xh	*on page 377*
FC0A_10C0	TCD6 Source Address (DMA_TCD6_SADDR)	32	R/W	0000_000Xh	*on page 366*
FC0A_10C4	TCD6 Transfer Attributes (DMA_TCD6_ATTR)	16	R/W	000Xh	*on page 367*
FC0A_10C6	TCD6 Signed Source Address Offset (DMA_TCD6_SOFF)	16	R/W	000Xh	*on page 368*
FC0A_10C8	(DMA_TCD6_NBYTES_MLNO)	32	R/W	0000_000Xh	*on page 369*
	(DMA_TCD6_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	*on page 369*
	(DMA_TCD6_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	*on page 370*
FC0A_10CC	(DMA_TCD6_SLAST)	32	R/W	0000_000Xh	*on page 371*
FC0A_10D0	(DMA_TCD6_DADDR)	32	R/W	0000_000Xh	*on page 372*

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_10D4	(DMA_TCD6_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD6 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD6_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_10D6	(DMA_TCD6_DOFF)	16	R/W	000Xh	on page 374
FC0A_10D8	(DMA_TCD6_DLASTSGA)	32	R/W	0000_000Xh	on page 375
FC0A_10DC	(DMA_TCD6_BITER_ELINKYES)	16	R/W	000Xh	on page 375
	(DMA_TCD6_BITER_ELINKNO)	16	R/W	000Xh	on page 376
FC0A_10DE	(DMA_TCD6_CSR)	16	R/W	000Xh	on page 377
FC0A_10E0	TCD7 Source Address (DMA_TCD7_SADDR)	32	R/W	0000_000Xh	on page 366
FC0A_10E4	TCD7 Transfer Attributes (DMA_TCD7_ATTR)	16	R/W	000Xh	on page 367
FC0A_10E6	TCD7 Signed Source Address Offset (DMA_TCD7_SOFF)	16	R/W	000Xh	on page 368
FC0A_10E8	(DMA_TCD7_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD7_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD7_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_10EC	(DMA_TCD7_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_10F0	(DMA_TCD7_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_10F4	(DMA_TCD7_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD7 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD7_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_10F6	(DMA_TCD7_DOFF)	16	R/W	000Xh	on page 374
FC0A_10F8	(DMA_TCD7_DLASTSGA)	32	R/W	0000_000Xh	on page 375
FC0A_10FC	(DMA_TCD7_BITER_ELINKYES)	16	R/W	000Xh	on page 375
	(DMA_TCD7_BITER_ELINKNO)	16	R/W	000Xh	on page 376
FC0A_10FE	(DMA_TCD7_CSR)	16	R/W	000Xh	on page 377
FC0A_1100	TCD8 Source Address (DMA_TCD8_SADDR)	32	R/W	0000_000Xh	on page 366
FC0A_1104	TCD8 Transfer Attributes (DMA_TCD8_ATTR)	16	R/W	000Xh	on page 367
FC0A_1106	TCD8 Signed Source Address Offset (DMA_TCD8_SOFF)	16	R/W	000Xh	on page 368

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1108	(DMA_TCD8_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD8_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD8_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_110C	(DMA_TCD8_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_1110	(DMA_TCD8_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_1114	(DMA_TCD8_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD8 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD8_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_1116	(DMA_TCD8_DOFF)	16	R/W	000Xh	on page 374
FC0A_1118	(DMA_TCD8_DLASTSGA)	32	R/W	0000_000Xh	on page 375
FC0A_111C	(DMA_TCD8_BITER_ELINKYES)	16	R/W	000Xh	on page 375
	(DMA_TCD8_BITER_ELINKNO)	16	R/W	000Xh	on page 376
FC0A_111E	(DMA_TCD8_CSR)	16	R/W	000Xh	on page 377
FC0A_1120	TCD9 Source Address (DMA_TCD9_SADDR)	32	R/W	0000_000Xh	on page 366
FC0A_1124	TCD9 Transfer Attributes (DMA_TCD9_ATTR)	16	R/W	000Xh	on page 367
FC0A_1126	TCD9 Signed Source Address Offset (DMA_TCD9_SOFF)	16	R/W	000Xh	on page 368
FC0A_1128	(DMA_TCD9_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD9_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD9_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_112C	(DMA_TCD9_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_1130	(DMA_TCD9_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_1134	(DMA_TCD9_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD9 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD9_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_1136	(DMA_TCD9_DOFF)	16	R/W	000Xh	on page 374
FC0A_1138	(DMA_TCD9_DLASTSGA)	32	R/W	0000_000Xh	on page 375
FC0A_113C	(DMA_TCD9_BITER_ELINKYES)	16	R/W	000Xh	on page 375
	(DMA_TCD9_BITER_ELINKNO)	16	R/W	000Xh	on page 376
FC0A_113E	(DMA_TCD9_CSR)	16	R/W	000Xh	on page 377
FC0A_1140	TCD10 Source Address (DMA_TCD10_SADDR)	32	R/W	0000_000Xh	on page 366

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_1144	TCD10 Transfer Attributes (DMA_TCD10_ATTR)	16	R/W	000Xh	on page 367
FC0A_1146	TCD10 Signed Source Address Offset (DMA_TCD10_SOFF)	16	R/W	000Xh	on page 368
FC0A_1148	(DMA_TCD10_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD10_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD10_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_114C	(DMA_TCD10_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_1150	(DMA_TCD10_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_1154	(DMA_TCD10_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD10 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD10_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_1156	(DMA_TCD10_DOFF)	16	R/W	000Xh	on page 374
FC0A_1158	(DMA_TCD10_DLASTSGA)	32	R/W	0000_000Xh	on page 375
FC0A_115C	(DMA_TCD10_BITER_ELINKYES)	16	R/W	000Xh	on page 375
	(DMA_TCD10_BITER_ELINKNO)	16	R/W	000Xh	on page 376
FC0A_115E	(DMA_TCD10_CSR)	16	R/W	000Xh	on page 377
FC0A_1160	TCD11 Source Address (DMA_TCD11_SADDR)	32	R/W	0000_000Xh	on page 366
FC0A_1164	TCD11 Transfer Attributes (DMA_TCD11_ATTR)	16	R/W	000Xh	on page 367
FC0A_1166	TCD11 Signed Source Address Offset (DMA_TCD11_SOFF)	16	R/W	000Xh	on page 368
FC0A_1168	(DMA_TCD11_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD11_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD11_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_116C	(DMA_TCD11_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_1170	(DMA_TCD11_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_1174	(DMA_TCD11_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD11 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD11_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_1176	(DMA_TCD11_DOFF)	16	R/W	000Xh	on page 374
FC0A_1178	(DMA_TCD11_DLASTSGA)	32	R/W	0000_000Xh	on page 375
FC0A_117C	(DMA_TCD11_BITER_ELINKYES)	16	R/W	000Xh	on page 375
	(DMA_TCD11_BITER_ELINKNO)	16	R/W	000Xh	on page 376

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_117E	(DMA_TCD11_CSR)	16	R/W	000Xh	on page 377
FC0A_1180	TCD12 Source Address (DMA_TCD12_SADDR)	32	R/W	0000_000Xh	on page 366
FC0A_1184	TCD12 Transfer Attributes (DMA_TCD12_ATTR)	16	R/W	000Xh	on page 367
FC0A_1186	TCD12 Signed Source Address Offset (DMA_TCD12_SOFF)	16	R/W	000Xh	on page 368
FC0A_1188	(DMA_TCD12_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD12_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD12_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_118C	(DMA_TCD12_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_1190	(DMA_TCD12_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_1194	(DMA_TCD12_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD12 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD12_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_1196	(DMA_TCD12_DOFF)	16	R/W	000Xh	on page 374
FC0A_1198	(DMA_TCD12_DLASTSGA)	32	R/W	0000_000Xh	on page 375
FC0A_119C	(DMA_TCD12_BITER_ELINKYES)	16	R/W	000Xh	on page 375
	(DMA_TCD12_BITER_ELINKNO)	16	R/W	000Xh	on page 376
FC0A_119E	(DMA_TCD12_CSR)	16	R/W	000Xh	on page 377
FC0A_11A0	TCD13 Source Address (DMA_TCD13_SADDR)	32	R/W	0000_000Xh	on page 366
FC0A_11A4	TCD13 Transfer Attributes (DMA_TCD13_ATTR)	16	R/W	000Xh	on page 367
FC0A_11A6	TCD13 Signed Source Address Offset (DMA_TCD13_SOFF)	16	R/W	000Xh	on page 368
FC0A_11A8	(DMA_TCD13_NBYTES_MLNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD13_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	on page 369
	(DMA_TCD13_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	on page 370
FC0A_11AC	(DMA_TCD13_SLAST)	32	R/W	0000_000Xh	on page 371
FC0A_11B0	(DMA_TCD13_DADDR)	32	R/W	0000_000Xh	on page 372
FC0A_11B4	(DMA_TCD13_CITER_ELINKYES)	16	R/W	000Xh	on page 372
	TCD13 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD13_CITER_ELINKNO)	16	R/W	000Xh	on page 373
FC0A_11B6	(DMA_TCD13_DOFF)	16	R/W	000Xh	on page 374

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_11B8	(DMA_TCD13_DLASTSGA)	32	R/W	0000_000Xh	*on page 375*
FC0A_11BC	(DMA_TCD13_BITER_ELINKYES)	16	R/W	000Xh	*on page 375*
	(DMA_TCD13_BITER_ELINKNO)	16	R/W	000Xh	*on page 376*
FC0A_11BE	(DMA_TCD13_CSR)	16	R/W	000Xh	*on page 377*
FC0A_11C0	TCD14 Source Address (DMA_TCD14_SADDR)	32	R/W	0000_000Xh	*on page 366*
FC0A_11C4	TCD14 Transfer Attributes (DMA_TCD14_ATTR)	16	R/W	000Xh	*on page 367*
FC0A_11C6	TCD14 Signed Source Address Offset (DMA_TCD14_SOFF)	16	R/W	000Xh	*on page 368*
FC0A_11C8	(DMA_TCD14_NBYTES_MLNO)	32	R/W	0000_000Xh	*on page 369*
	(DMA_TCD14_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	*on page 369*
	(DMA_TCD14_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	*on page 370*
FC0A_11CC	(DMA_TCD14_SLAST)	32	R/W	0000_000Xh	*on page 371*
FC0A_11D0	(DMA_TCD14_DADDR)	32	R/W	0000_000Xh	*on page 372*
FC0A_11D4	(DMA_TCD14_CITER_ELINKYES)	16	R/W	000Xh	*on page 372*
	TCD14 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD14_CITER_ELINKNO)	16	R/W	000Xh	*on page 373*
FC0A_11D6	(DMA_TCD14_DOFF)	16	R/W	000Xh	*on page 374*
FC0A_11D8	(DMA_TCD14_DLASTSGA)	32	R/W	0000_000Xh	*on page 375*
FC0A_11DC	(DMA_TCD14_BITER_ELINKYES)	16	R/W	000Xh	*on page 375*
	(DMA_TCD14_BITER_ELINKNO)	16	R/W	000Xh	*on page 376*
FC0A_11DE	(DMA_TCD14_CSR)	16	R/W	000Xh	*on page 377*
FC0A_11E0	TCD15 Source Address (DMA_TCD15_SADDR)	32	R/W	0000_000Xh	*on page 366*
FC0A_11E4	TCD15 Transfer Attributes (DMA_TCD15_ATTR)	16	R/W	000Xh	*on page 367*
FC0A_11E6	TCD15 Signed Source Address Offset (DMA_TCD15_SOFF)	16	R/W	000Xh	*on page 368*
FC0A_11E8	(DMA_TCD15_NBYTES_MLNO)	32	R/W	0000_000Xh	*on page 369*
	(DMA_TCD15_NBYTES_MLOFFNO)	32	R/W	0000_000Xh	*on page 369*
	(DMA_TCD15_NBYTES_MLOFFYES)	32	R/W	0000_000Xh	*on page 370*
FC0A_11EC	(DMA_TCD15_SLAST)	32	R/W	0000_000Xh	*on page 371*
FC0A_11F0	(DMA_TCD15_DADDR)	32	R/W	0000_000Xh	*on page 372*

Table 163. DMA memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Location
FC0A_11F4	(DMA_TCD15_CITER_ELINKYES)	16	R/W	000Xh	*on page 372*
	TCD15 Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD15_CITER_ELINKNO)	16	R/W	000Xh	*on page 373*
FC0A_11F6	(DMA_TCD15_DOFF)	16	R/W	000Xh	*on page 374*
FC0A_11F8	(DMA_TCD15_DLASTSGA)	32	R/W	0000_000Xh	*on page 375*
FC0A_11FC	(DMA_TCD15_BITER_ELINKYES)	16	R/W	000Xh	*on page 375*
	(DMA_TCD15_BITER_ELINKNO)	16	R/W	000Xh	*on page 376*
FC0A_11FE	(DMA_TCD15_CSR)	16	R/W	000Xh	*on page 377*
FC0A_1200–FC0A_13FF	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).

Address: FC0A_0000 + 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								EMLM	CLM	HALT	HOE	ERGA	ERCA	EDBG	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 119. Control Register (DMA_CR)

Table 164. 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.
16–23 Reserved	This read-only field is reserved and always has the value zero.
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.
29 ERCA	Enable round-robin channel arbitration 0 Fixed-priority arbitration is used for channel selection. 1 Round-robin arbitration is used for channel selection.

Table 164. DMA_CR field descriptions(Continued)

Field	Description
30 EDBG	Enable debug 0 When in Debug mode the DMA continues to operate. 1 When in Debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system or the EDBG bit is cleared.
31 Reserved	This bit is reserved and always has the value zero.

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: Error reporting and handling*, for more details.

Address: FC0A_0000 + 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 120. Error Status Register (DMA_ES)

Table 165. 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 0No uncorrectable ECC error 1The last recorded error was an uncorrectable TCD RAM error
15 ECX	Transfer canceled 0 No canceled transfers 1 The last recorded entry was a canceled transfer by the error cancel transfer input
16 Reserved	This read-only bitfield is reserved and always has the value zero.

Table 165. DMA_ES field descriptions(Continued)

Field	Description
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 and/or last recorded error canccanceledelled 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 Low (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.

Address: FC0A_0000 + 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	ERQ 15	ERQ 14	ERQ 13	ERQ 12	ERQ 11	ERQ 10	ERQ 9	ERQ 8	ERQ 7	ERQ 6	ERQ 5	ERQ 4	ERQ 3	ERQ 2	ERQ 1	ERQ 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 121. Enable Request Register Low (DMA_ERQL)

Table 166. 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
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

Table 166. DMA_ERQL field descriptions(Continued)

Field	Description
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 Low (DMA_EEIL)

The Enable Error Interrupt Register Low (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.

Address: FC0A_0000 + 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	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 122. Enable Error Interrupt Register Low (DMA_EEIL)

Table 167. 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
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

Table 167. DMA_EEIL field descriptions(Continued)

Field	Description
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.

Address: FC0A_0000 + 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 123. Set Enable Request Register (DMA_SERQ)

Table 168. DMA_SERQ field descriptions

Field	Description
0 NOP	0Normal operation 1No operation, ignore bits 1–7 of this register

Table 168. DMA_SERQ field descriptions(Continued)

Field	Description
1 SAER	Set all enable requests 0Set only those ERQ bits specified in the SERQ field 1Set 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.

Address: FC0A_0000 + 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 124. Clear Enable Request Register (DMA_CERQ)

Table 169. DMA_CERQ field descriptions

Field	Description
0 NOP	0Normal operation 1No operation, ignore bits 1–7 of this register
1 CAER	Clear all enable requests 0Clear only those ERQ bits specified in the CERQ field 1Clear 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.

Address: FC0A_0000 + 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 125. Set Enable Error Interrupt Register (DMA_SEEI)

Table 170. DMA_SEEI field descriptions

Field	Description
0 NOP	0Normal operation 1No operation, ignore bits 1–7 of this register
1 SAEE	Sets all enable error interrupts 0Set only those EEI bits specified in the SEEI field. 1Sets 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.

Address: FC0A_0000 + 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 126. Clear Enable Error Interrupt Register (DMA_CEEI)

Table 171. DMA_CEEI field descriptions

Field	Description
0 NOP	0Normal operation 1No operation, ignore bits 1–7 of this register
1 CAEE	Clear all enable error interrupts 0Clear only those EEI bits specified in the CEEI field 1Clear 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.

Address: FC0A_0000 + 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 127. Clear Interrupt Request Register (DMA_CINT)

Table 172. DMA_CINT field descriptions

Field	Description
0 NOP	0Normal operation 1No operation, ignore bits 1–7 of this register
1 CAIR	Clear all interrupt requests 0Clear only those INT bits specified in the CINT field 1Clear 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.

Address: FC0A_0000 + 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 128. Clear Error Register (DMA_CERR)

Table 173. DMA_CERR field descriptions

Field	Description
0 NOP	0Normal operation 1No operation, ignore bits 1–7 of this register
1 CAEI	Clear all error indicators 0Clear only those ERR bits specified in the CERR field 1Clear 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.

Address: FC0A_0000 + 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 129. Set START Bit Register (DMA_SSRT)

Table 174. DMA_SSRT field descriptions

Field	Description
0 NOP	0Normal operation 1No operation, ignore bits 1–7 of this register
1 SAST	Set all START bits (activates all channels) 0Set only those TCD*n*_CSR[START] bits specified in the SSRT field 1Set 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.

Address: FC0A_0000 + 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 130. Clear DONE Status Bit Register (DMA_CDNE)

Table 175. DMA_CDNE field descriptions

Field	Description
0 NOP	0Normal operation 1No operation, ignore bits 1–7 of this register
1 CADN	Clears all DONE bits 0Clears only those TCD*n*_CSR[DONE] bits specified in the CDNE field 1Clears 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 Low (DMA_INTL)

The Interrupt Request Register Low (DMA_INTL) provides a bit map for the lower half of 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.

Address: FC0A_0000 + 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	INT 15	INT 14	INT 13	INT 12	INT 11	INT 10	INT 9	INT 8	INT 7	INT 6	INT 5	INT 4	INT 3	INT 2	INT 1	INT 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 131. Interrupt Request Register Low (DMA_INTL)

Table 176. 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 176. 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 Low (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 across groups of 16 channels to form several group error interrupt requests that are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: FC0A_0000 + 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	ERR 15	ERR 14	ERR 13	ERR 12	ERR 11	ERR 10	ERR 9	ERR 8	ERR 7	ERR 6	ERR 5	ERR 4	ERR 3	ERR 2	ERR 1	ERR 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 132. Error Register Low (DMA_ERRL)

Table 177. 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 177. 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

Note: These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ{H,L} bits.

19.3.15 Hardware Request Status Register Low (DMA_HRSL)

The HRSL provides a bit map for the DMA channels, signaling the presence of a qualified hardware service request for each channel. HRSL covers channels 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.

Address: FC0A_0000 + 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	HRS 15	HRS 14	HRS 13	HRS 12	HRS 11	HRS 10	HRS 9	HRS 8	HRS 7	HRS 6	HRS 5	HRS 4	HRS 3	HRS 2	HRS 1	HRS 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 133. Hardware Request Status Register Low (DMA_HRSL)

Table 178. 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 178. 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.

Address: FC0A_0000h base + 100h offset + (1d × *n*), where n = 0d to 63d 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

Figure 134. Channel *n* Priority Register (DMA_DCHPRI*n*)

1. x = Undefined at reset.

19.3.17 Channel *n* Master ID Register (DMA_DCHMID*n*)

Table 179. 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 for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, i.e., DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.

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.

Address: FC0A_0000h base + 140h offset + (1d × n), where n = 0d to 63d Access: User read/write

	0	1	2	3	4	5	6	7
R	EMI	PAL	0		MID			
W								
Reset	0	0	0	0	0	0	0	0

Figure 135. Channel *n* Master ID Register (DMA_DCHMID*n*)

Table 180. 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)

See TCD Source Address register figure and DMA_TCD*n*_SADDR field descriptions table as follows.

Address: FC0A_0000h base + 1000h offset + (32d × n), where n = 0d to 63d Access: User read/write

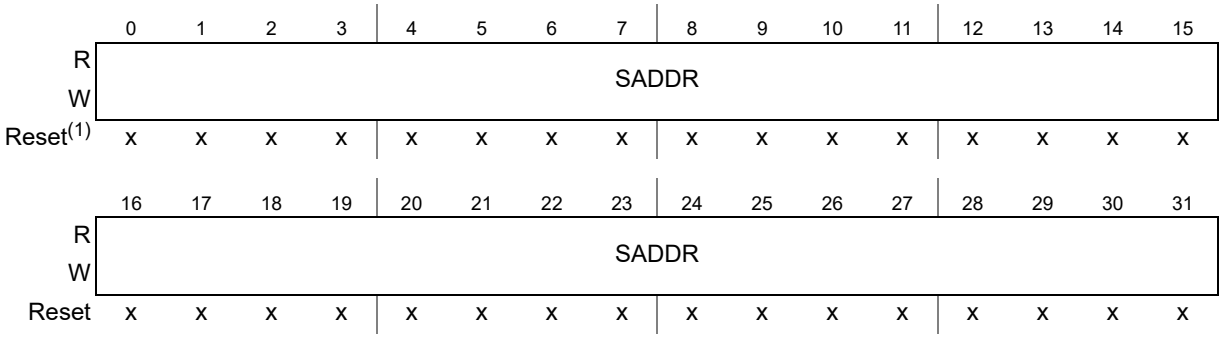


Figure 136. TCD Source Address (DMA_TCD*n*_SADDR)

1. x = Undefined at reset.

Table 181. 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)

See TCD Transfer Attributes register figure and DMA_TCD*n*_ATTR field descriptions table as follows.

Address: FC0A_0000h base + 1004h offset + (32d × n), where n = 0d to 63d Access: User read/write

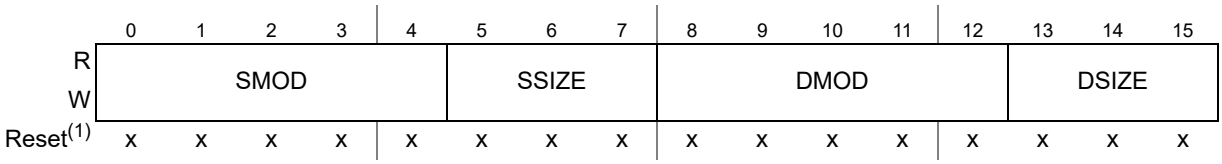


Figure 137. TCD Transfer Attributes (DMA_TCD*n*_ATTR)

1. x = Undefined at reset.

Table 182. DMA_TCD*n*_ATTR field descriptions

Field	Description
16–20 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.
21–23 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
24–28 DMOD	Destination address modulo See the SMOD definition
29–31 DSIZE	Destination data transfer size See the SSIZE definition

19.3.20 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

See TCD Signed Source Address Offset register figure and DMA_TCD*n*_SOFF field descriptions table as follows.

Address: FC0A_0000h base + 1006h offset + (32d × n), where n = 0d to 63d 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

Figure 138. TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

1. x = Undefined at reset.

Table 183. DMA_TCD*n*_SOFF field descriptions

Field	Description
16–31 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.

Address: FC0A_0000h base + 1008h offset + (32d × n), where n = 0d to 63d

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

Figure 139. TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD*n*_NBYTES_MLNO)

1. x = Undefined at reset.

Table 184. 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

Address: FC0A_0000h base + 1008h offset + (32d × n), where n = 0d to 63d Access: User read/write

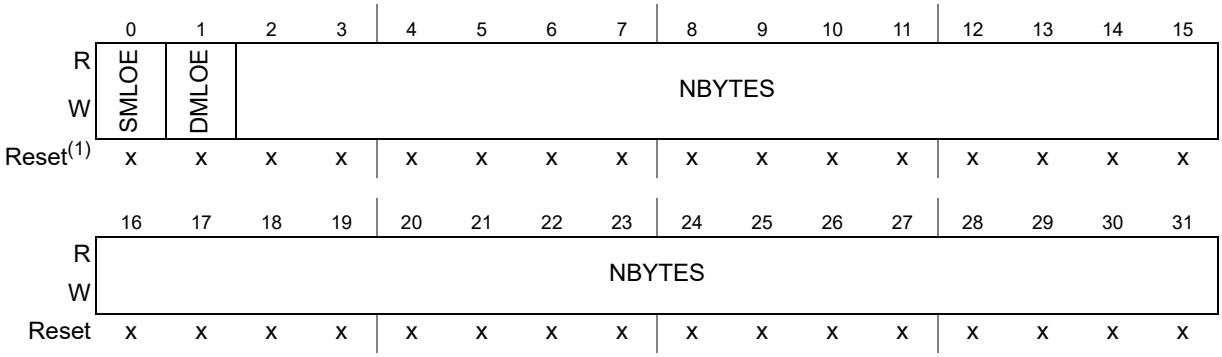


Figure 140. TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO)

1. x = Undefined at reset.

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)

Address: FC0A_0000h base + 1008h offset + (32d × n), where n = 0d to 63d Access: User read/write

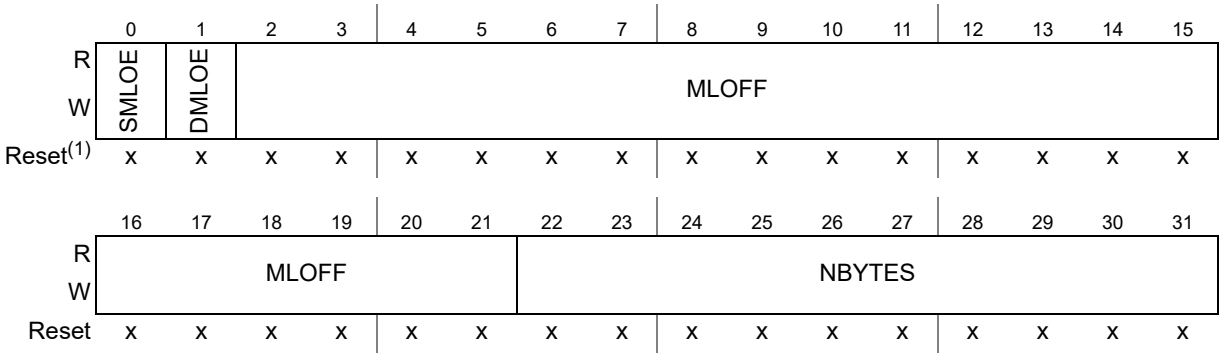


Figure 141. TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES)

1. x = Undefined at reset.

Table 185. 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)

Address: FC0A_0000h base + 100Ch offset + (32d × n), where n = 0d to 63d

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

Figure 142. TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)

1. x = Undefined at reset.

Table 186. 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)

See TCD Destination Address register figure and DMA_TCD*n*_DADDR field descriptions table as follows.

Address: FC0A_0000h base + 1010h offset + (32d × n), where n = 0d to 63d 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

Figure 143. TCD Destination Address (DMA_TCD*n*_DADDR)

1. x = Undefined at reset.

Table 187. 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)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) register figure and DMA_TCDn_CITER_ELINKYES field descriptions table as follows.

Address: FC0A_0000h base + 1014h offset + (32d × n), where n = 0d to 63d Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	E LINK	LINKCH							CITER							
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 144. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES)

1. x = Undefined at reset.

Table 188. DMA_TCD*n*_CITER_ELINKYES field descriptions

Field	Description
16 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
17–22 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.
23–31 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 (e.g., 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)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) register figure and DMA_TCDn_CITER_ELINKNO field descriptions table as follows.

Address: FC0A_0000h base + 1014h offset + (32d × n), where n = 0d to 63d 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

Figure 145. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)

1. x = Undefined at reset.

Table 189. DMA_TCD*n*_CITER_ELINKNO field descriptions

Field	Description
16 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
17–31 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)

See TCD Signed Destination Address Offset register figure and DMA_TCD*n*_DOFF field descriptions table as follows.

Address: FC0A_0000h base + 1016h offset + (32d × n), where n = 0d to 63d 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

Figure 146. TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)

1. x = Undefined at reset.

Table 190. DMA_TCD*n*_DOFF field descriptions

Field	Description
16–31 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)

Address: FC0A_0000h base + 1018h offset + (32d × n), where n = 0d to 63d 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

Figure 147. TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)

1. x = Undefined at reset.

Table 191. 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)

Address: FC0A_0000h base + 101Ch offset + (32d × n), where n = 0d to 63d Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ELINK	LINKCH							BITER							
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 148. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)

1. x = Undefined at reset.

Table 192. DMA_TCD*n*_BITER_ELINKYES field descriptions

Field	Description
16 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
17–22 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.
23–31 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)

Address: FC0A_0000h base + 101Ch offset + (32d × n), where n = 0d to 63d Access: User read/write

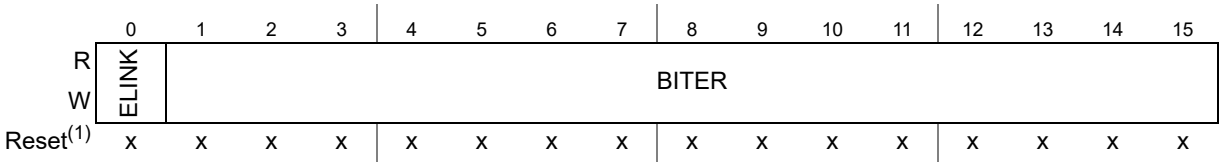


Figure 149. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)

1. x = Undefined at reset.

Table 193. DMA_TCD*n*_BITER_ELINKNO field descriptions

Field	Description
16 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
17–31 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)

See TCD Control and Status register figure and DMA_TCD*n*_CSR field descriptions table as follows.

Address: FC0A_0000h base + 101Eh offset + (32d × n), where n = 0d to 63d 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

Figure 150. TCD Control and Status (DMA_TCD*n*_CSR)

1. x = Undefined at reset.

Table 194. DMA_TCD*n*_CSR field descriptions

Field	Description
16–17 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 (XBAR). **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

18–23
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.

24 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.
25 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.
26 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
27 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.

Table 194. DMA_TCD*n*_CSR field descriptions(Continued)

Field	Description
28 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
29 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
30 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
31 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:

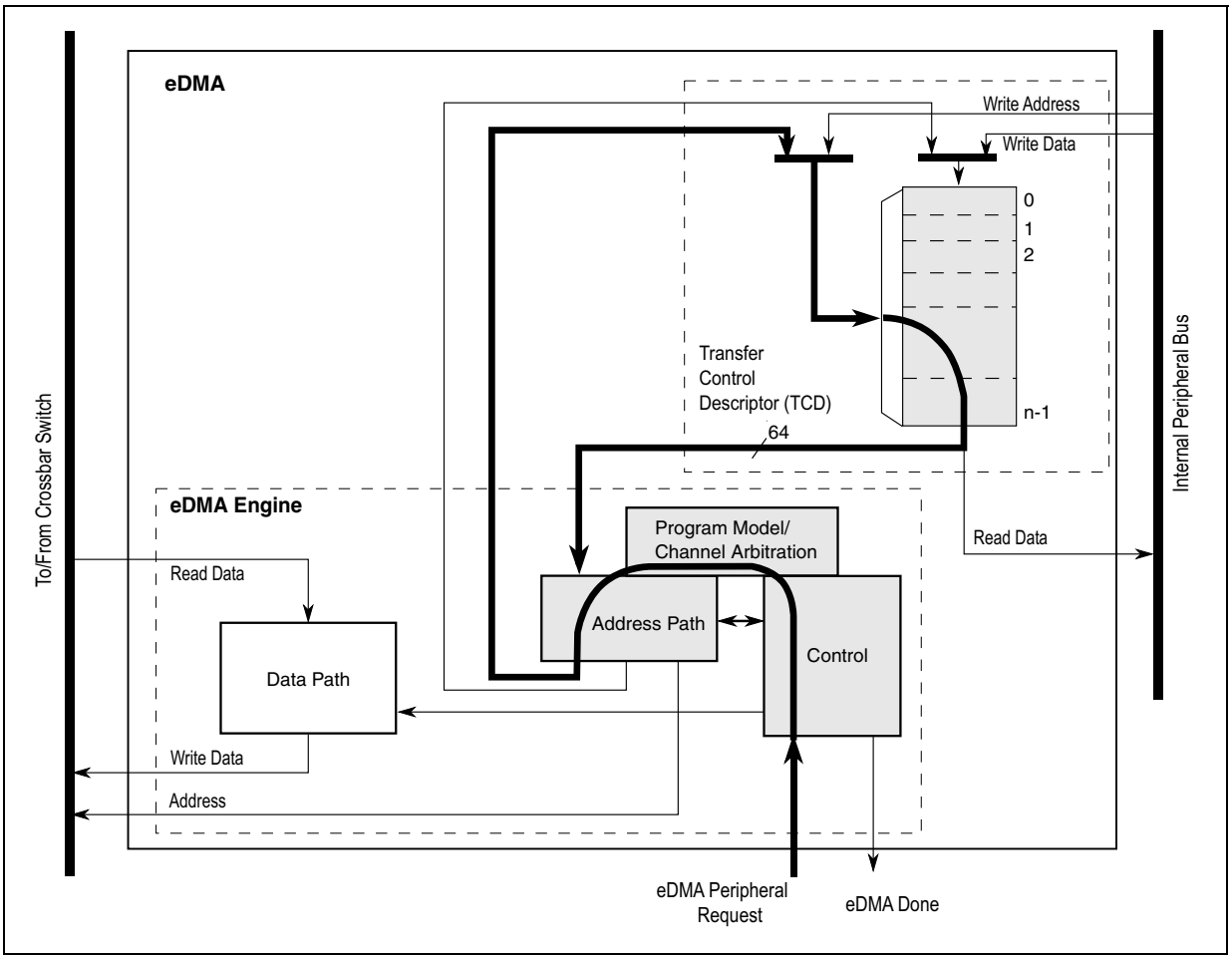
    TCD storage is implemented using a single-port, synchronous RAM array.

19.4.2 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments. As shown in *Figure 151*, 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

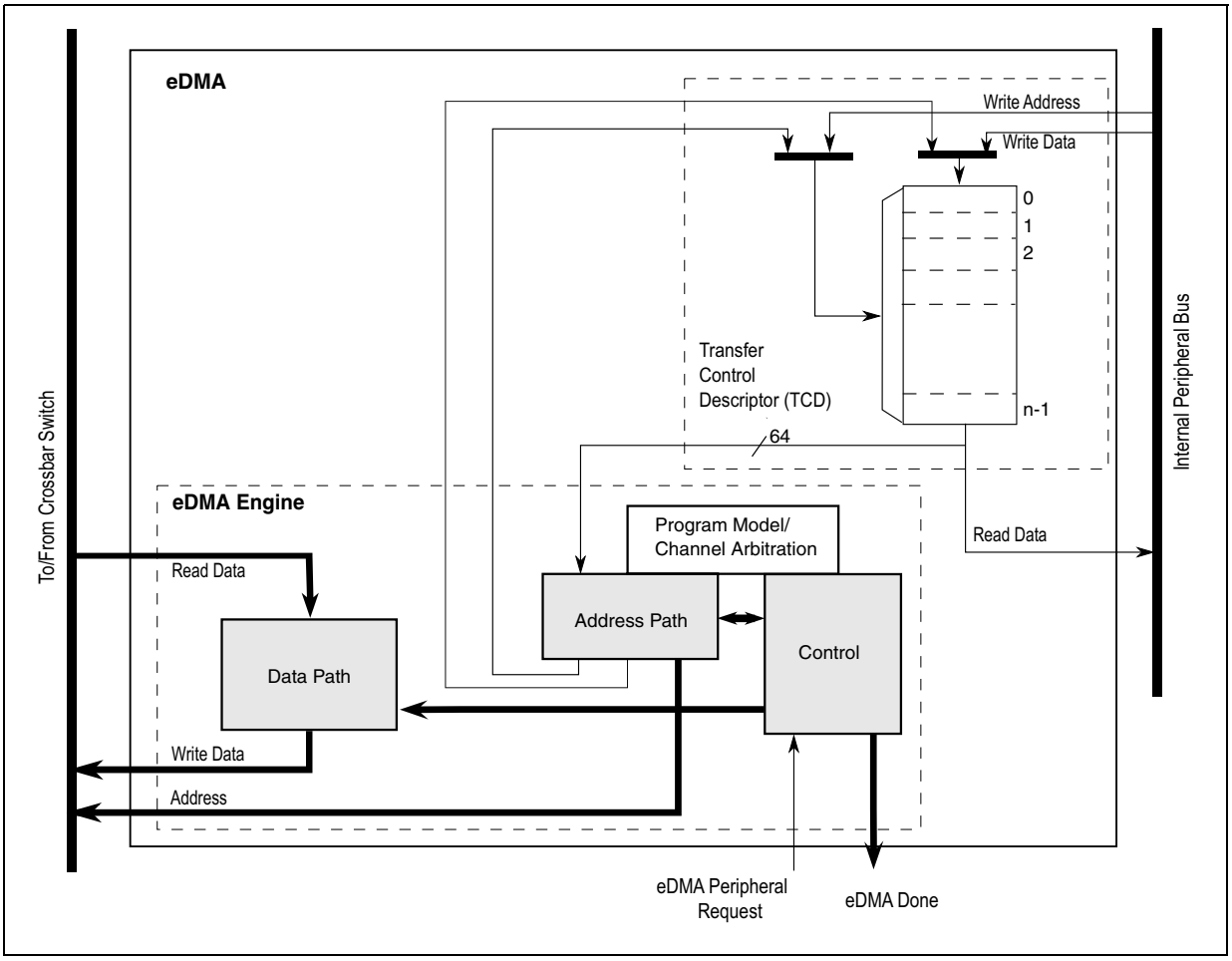
memory for TCD*n*. Next, the TCD memory is accessed and the descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

Figure 151. DMA operation, part 1



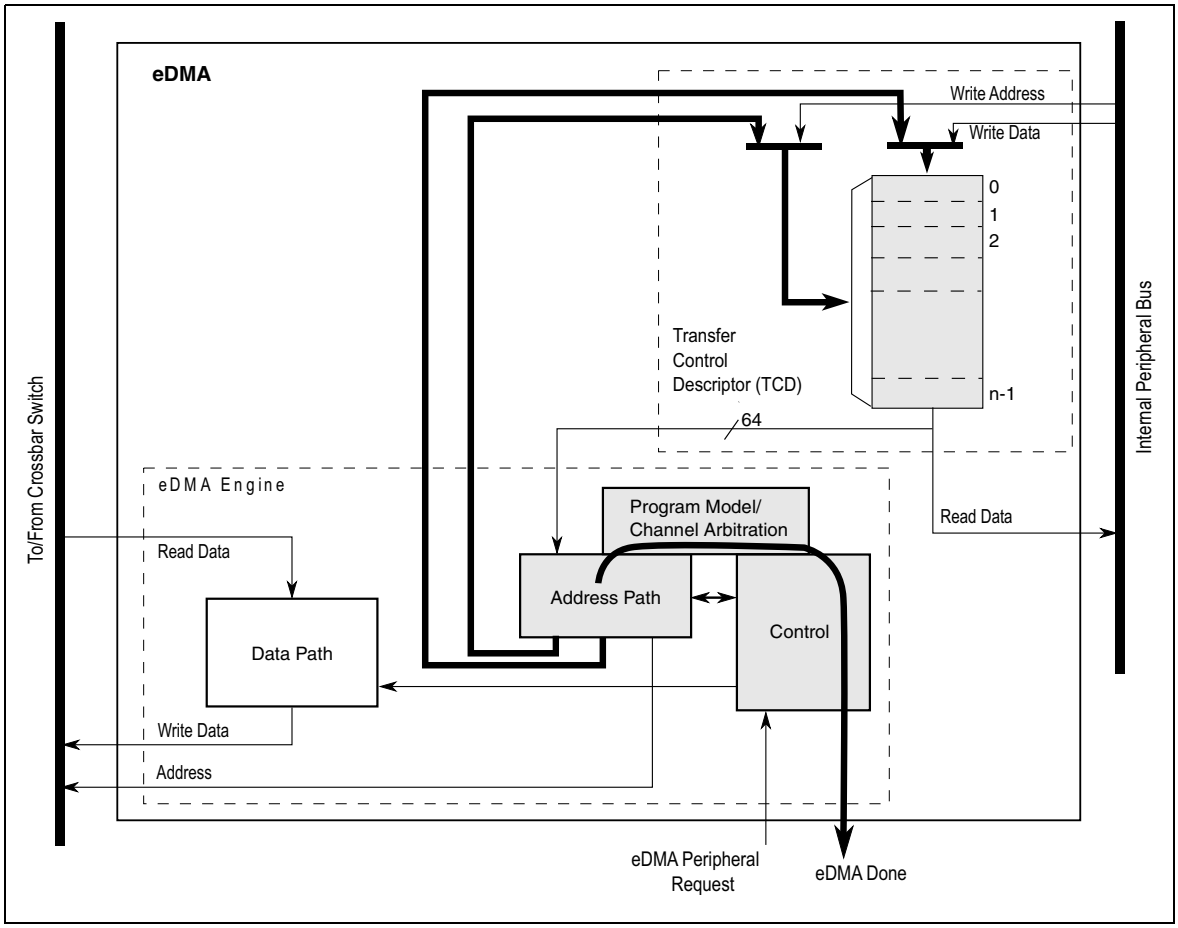
In the second part of the basic data flow as shown in *Figure 152* the modules associated with the data transfer (address path, data path, and control) sequence through source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily 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 152. 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 153*.

Figure 153. 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 canceled 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 canceled channel number and ECX is set. The TCD of a canceled 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 canceled 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 195* and *Table 196*. The following assumptions apply to those tables:

- Internal SRAM can be accessed with zero wait states when viewed from the system bus data phase.
- All internal peripheral bus reads require one wait-state, and internal peripheral bus writes one wait-state, again viewed from the system bus data phase.
- All internal peripheral bus accesses are 32 bits in size.
- All internal SRAM accesses are 64 bits in size.

Table 195 presents a peak transfer rate comparison, measured in megabytes per second.

Table 195. eDMA peak transfer rates (Mbyte/s)

System speed, width	From internal SRAM to internal SRAM	From 32-bit internal peripheral bus to internal SRAM	From internal SRAM to 32-bit internal peripheral bus
100 MHz, 64 bits	400.0	160.0	160.0

Table 195. 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[(d)]: 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.

d. "*x*" represents an undefined cycle value dependant on transfer sizes and response times for the channel's read and write accesses.

For a single SRAM read and two 64-bit internal peripheral bus writes, the combined data phase time is five cycles.

- Cycle $x^{(d)}$ + 1: This cycle represents the data phase of the last destination write.
- Cycle $x^{(d)}$ + 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 $x^{(d)}$ + 3: The appropriate fields in the first part of the TCD*n* are written back into the local memory.
- Cycle $x^{(d)}$ + 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 $x^{(d)}$ + 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 196*. 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 196. eDMA peak request rate (MReq/s)

System frequency (MHz)	Request rate	
	zero wait states	with wait states
100.0	11.1	10.0
150.0	16.6	15.0
200.0	22.2	20.0

A general formula to compute the peak request rate (with overlapping requests) is:

Equation 1 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 2 PEAKreq = 150 MHz / [4 + (1 + 0) + (1 + 1) + 3] cycles = 15.0 Mreq/s

For an internal peripheral bus-to-SRAM transfer,

Equation 3 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 197*) 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 197. TCD control and status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

Figure 154 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 154. Example of multiple loop iterations

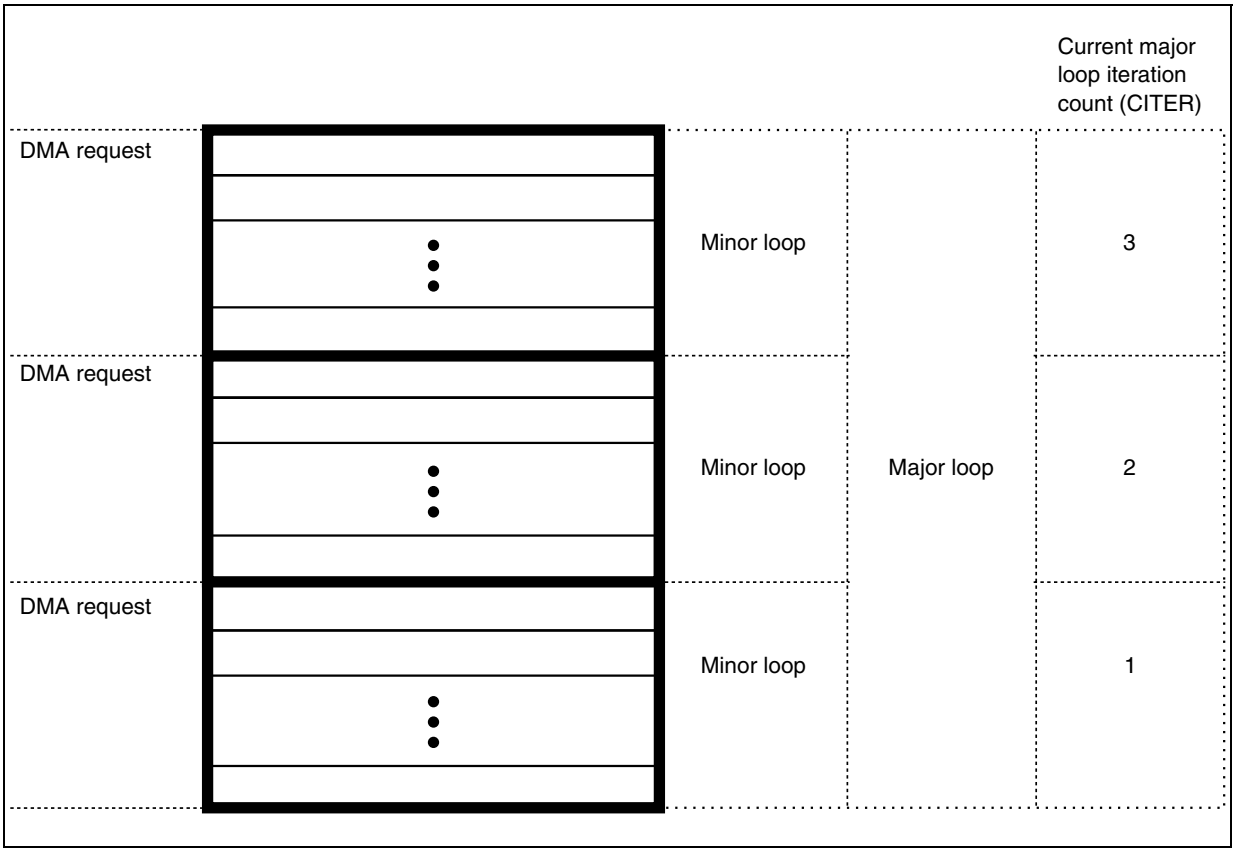
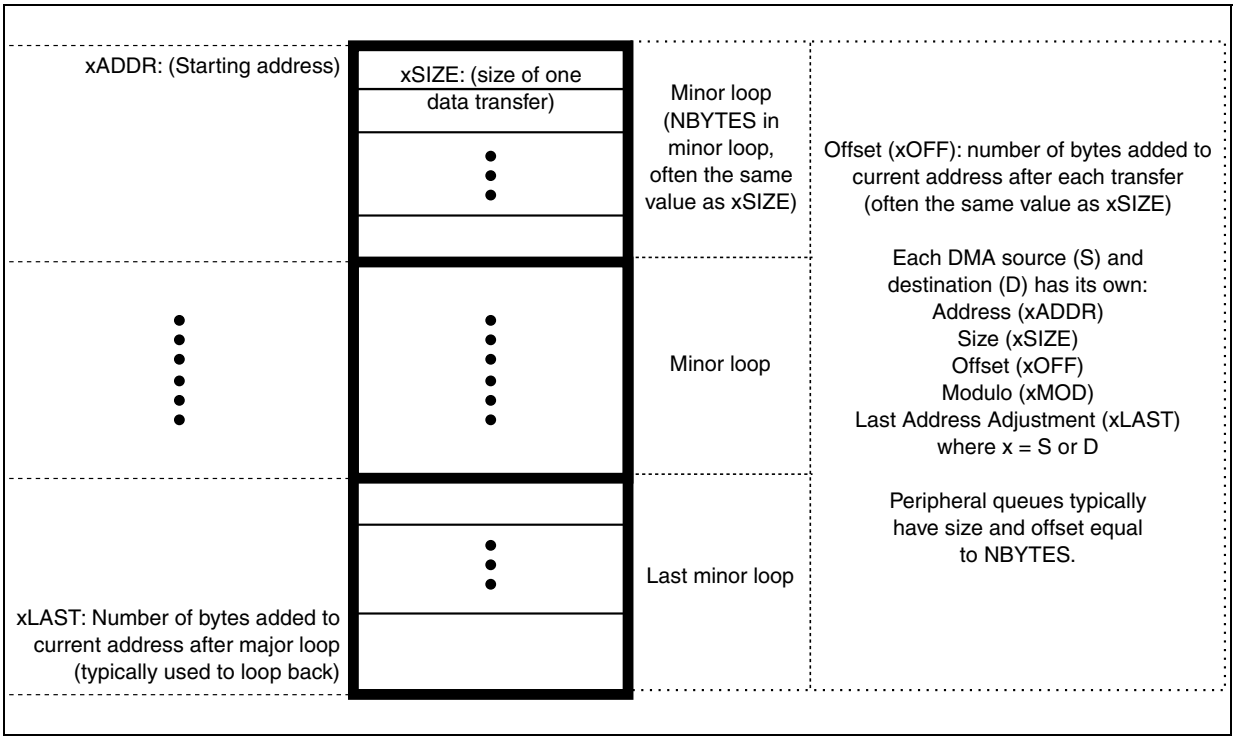


Figure 155 lists the memory array terms and how the TCD settings interrelate.

Figure 155. 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 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 198 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 198. 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 199. TCD status sequence – software activated channel

Step	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing

Table 199. TCD status sequence – software activated channel(Continued)

Step	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware (peripheral) initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute a sequence for a hardware-activated channel as shown in the following table.

Table 200. 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 201 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 201. Channel linking parameters

Desired link behavior	TCD control field name	Description
Link at end of minor loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH] Link channel number when linking at end of minor loop (current iteration)	

Table 201. Channel linking parameters(Continued)

Desired link behavior	TCD control field name	Description
Link at end of major loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH] Link channel number when linking at end of major loop	

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 202* is recommended when executing a dynamic channel link request.

Table 202. 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 203* 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 203. Coherency model for method 1

Step	Action
1	When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2	Write 1b to theTCD.d_req bit. **Note:** Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.
3	Write theTCD.dlast_sga field with the scatter/gather address.
4	Write 1b to the TCD.e_sg bit.

Table 203. Coherency model for method 1(Continued)

Step	Action
5	Read back the 16-bit TCD control/status field.
6	Test the TCD.e_sg request status and TCD.major.linkch value: – If e_sg = 1b, the dynamic link attempt was successful. – If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring). – If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

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 204* may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD identification (ID).

Table 204. Coherency model for method 2

Step	Action
1	Write 1b to theTCD.d_req bit. **Note:** Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.
2	Write theTCD.dlast_sga field with the scatter/gather address.
3	Write 1b to the TCD.e_sg bit.
4	Read back the TCD.e_sg bit.
5	Test the TCD.e_sg request status: – If e_sg = 1b, the dynamic link attempt was successful. – If e_sg = 0b, read the 32-bit TCD dlast_sga field. – If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring). – If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

20 Direct Memory Access Multiplexer (DMAMUX)

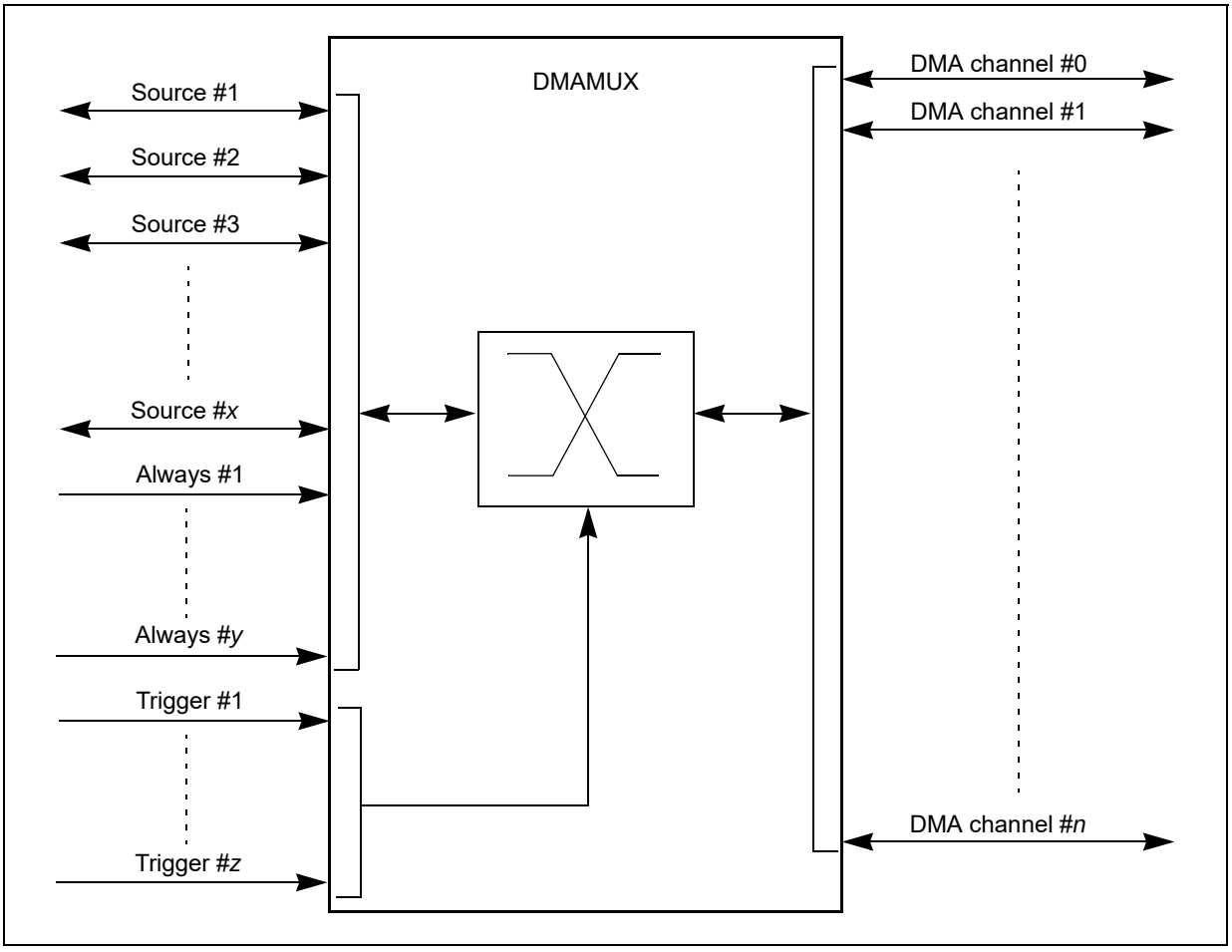
20.1 Introduction

Note: For the chip-specific implementation details of this module's instances, see the chip configuration information.

20.1.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 16 DMA channels. This process is illustrated in *Figure 156*.

Figure 156. DMAMUX block diagram



20.1.2 Features

The DMAMUX module provides these features:

- Up to 64 peripheral slots and up to six always-on slots can be routed to 16 channels.
- 16 independently selectable DMA channel routers.
  - The first eight channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of the possible peripheral DMA slots or to one of the always-on slots.

20.1.3 Modes of operation

The following operating modes are available:

- Disabled mode

  In this mode, the DMA channel is disabled. Because 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 DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place, for example, changing the period of a DMA trigger.
- Normal mode

  In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX 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 available only for channels 0–5.

20.2 External signal description

The DMAMUX has no external pins.

20.3 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

Table 205. DMAMUX memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Channel Configuration register (DMAMUX_CHCFG0)	8	R/W	00h	*20.3.1/402*
1	Channel Configuration register (DMAMUX_CHCFG1)	8	R/W	00h	*20.3.1/402*
2	Channel Configuration register (DMAMUX_CHCFG2)	8	R/W	00h	*20.3.1/402*
3	Channel Configuration register (DMAMUX_CHCFG3)	8	R/W	00h	*20.3.1/402*
4	Channel Configuration register (DMAMUX_CHCFG4)	8	R/W	00h	*20.3.1/402*

Table 205. DMAMUX memory map(Continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5	Channel Configuration register (DMAMUX_CHCFG5)	8	R/W	00h	*20.3.1/402*
6	Channel Configuration register (DMAMUX_CHCFG6)	8	R/W	00h	*20.3.1/402*
7	Channel Configuration register (DMAMUX_CHCFG7)	8	R/W	00h	*20.3.1/402*
8	Channel Configuration register (DMAMUX_CHCFG8)	8	R/W	00h	*20.3.1/402*
9	Channel Configuration register (DMAMUX_CHCFG9)	8	R/W	00h	*20.3.1/402*
A	Channel Configuration register (DMAMUX_CHCFG10)	8	R/W	00h	*20.3.1/402*
B	Channel Configuration register (DMAMUX_CHCFG11)	8	R/W	00h	*20.3.1/402*
C	Channel Configuration register (DMAMUX_CHCFG12)	8	R/W	00h	*20.3.1/402*
D	Channel Configuration register (DMAMUX_CHCFG13)	8	R/W	00h	*20.3.1/402*
E	Channel Configuration register (DMAMUX_CHCFG14)	8	R/W	00h	*20.3.1/402*
F	Channel Configuration register (DMAMUX_CHCFG15)	8	R/W	00h	*20.3.1/402*

20.3.1 Channel Configuration register (DMAMUX_CHCFGn)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

Note: Setting multiple CHCFG registers with the same source value will result in unpredictable behavior.

Note: Before changing the trigger or source settings, a DMA channel must be disabled via CHCFGn[ENBL].

Address: 0h base + 0h offset + (1d × i), where i=0d to 15d 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 157. Channel Configuration register (DMAMUX_CHCFGn)

Table 206. DMAMUX_CHCFGn field descriptions

Field	Description
0 ENBL	DMA Channel Enable. Enables the DMA channel 0 DMA channel is disabled. This mode is primarily used during configuration of the DMAMUX. 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 . Enables the periodic trigger capability for the triggered DMA Channel. 0 Triggering is disabled. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode) 1 Triggering is enabled. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode.
2–7 SOURCE	DMA Channel Source (Slot) Specifies which DMA source, if any, is routed to a particular DMA channel. See your device's chip configuration details for information about the peripherals and their slot numbers.

20.4 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels. As such, configuration of the DMAMUX 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 DMAMUX may be changed during the normal operation of the system.

Functionally, the DMAMUX channels may be divided into two classes:

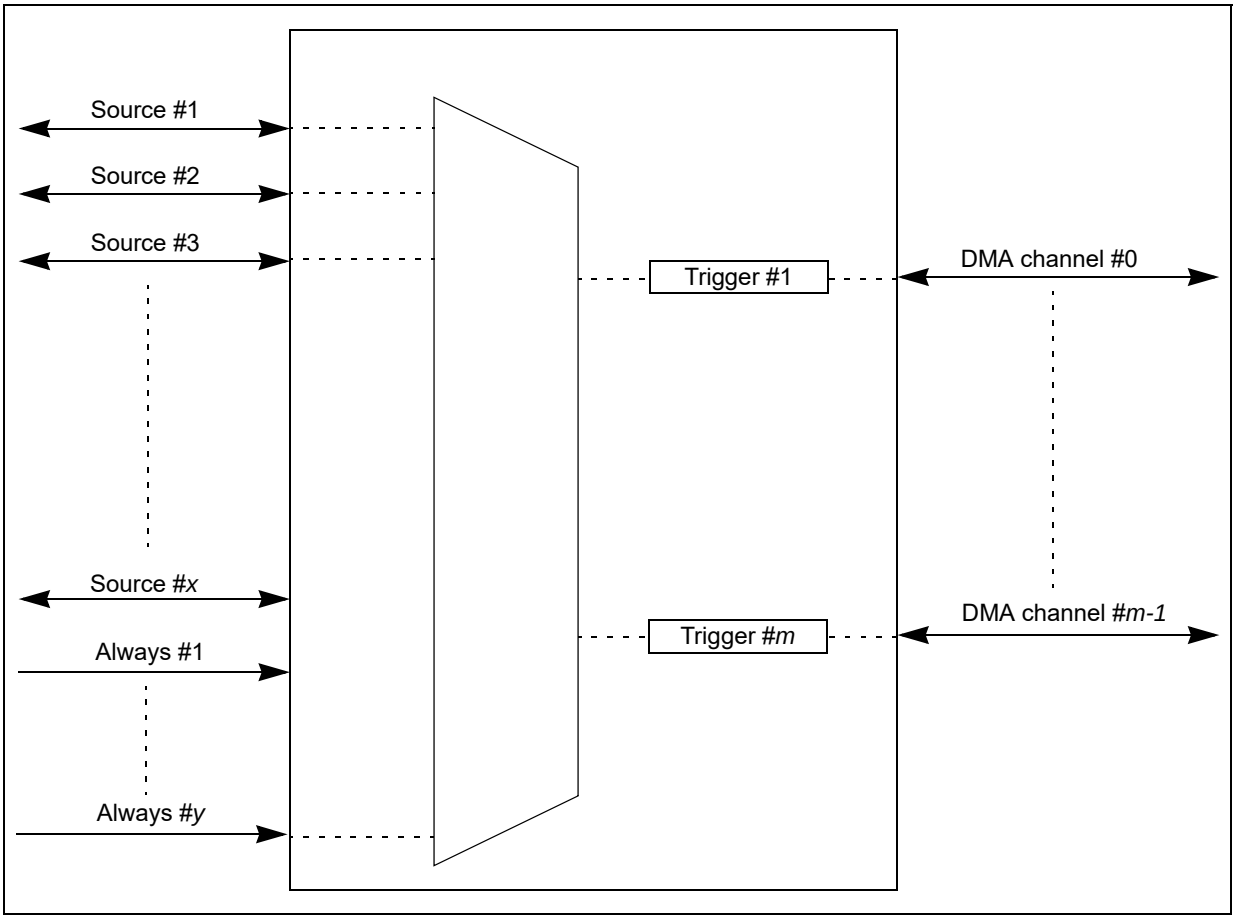
- Channels that implement the normal routing functionality plus periodic triggering capability
- Channels that implement only the normal routing functionality

20.4.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first channels of the DMAMUX 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. See the section on periodic interrupt timer for more information on this topic.

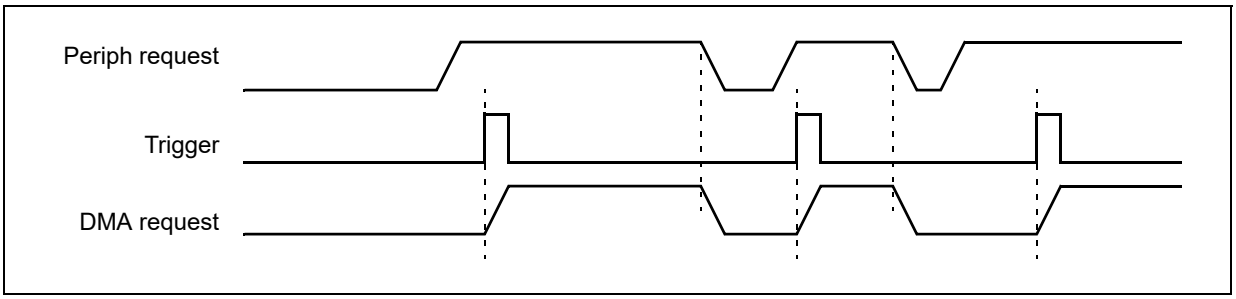
Note: Because of the dynamic nature of the system (due to DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

Figure 158. DMAMUX triggered channels



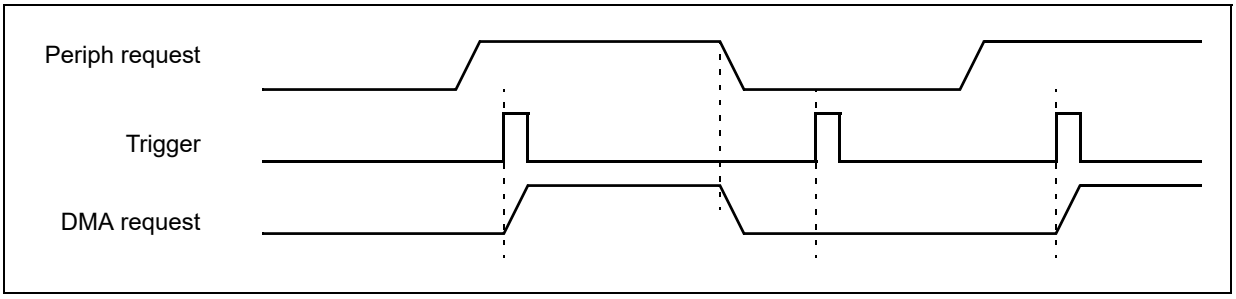
The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event has been seen. This is illustrated in *Figure 159*.

Figure 159. DMAMUX channel triggering: normal operation



After 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,then that trigger will be ignored. This situation is illustrated in *Figure 160*.

Figure 160. DMAMUX 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. After it has been set up, 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, including resolution, range of values, and so on, may be found in the periodic interrupt timer section.

20.4.2 DMA channels with no triggering capability

The other channels of the DMAMUX 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 six additional DMA sources that are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the sources that are always

enabled provide no such “throttling” of the data transfers. These sources are most useful in the following cases:

- Performing 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).
- Performing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing 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, an 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 that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are:

- Transfer all data in a single minor loop.

  By configuring the DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.
- 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 must be disabled in the DMA channel MUX.
- Use an always-enabled DMA source.

  In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX 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

This section provides instructions for initializing the DMA channel MUX.

20.5.1 Reset

The reset state of each individual bit is shown in *Section 20.3: Memory map/register definition*. 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 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.

To configure source #5 transmit for use with DMA channel 1, with periodic triggering capability:

1. Write 0x00 to CHCFG1 (base address + 0x01).
2. Configure channel 1 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCFG1 (base address + 0x01).

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"
         :
         :
*CHCFG1 = 0x00;
*CHCFG1 = 0xC5;
```

To enable a source without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 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.

To configure source #5 transmit for use with DMA channel 1, with no periodic triggering capability:

1. Write 0x00 to CHCFG1 (base address + 0x01).
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1 (base address + 0x01).

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"
         :
         :
*CHCFG1 = 0x00;
*CHCFG1 = 0x85;
```

To disable 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. See the appropriate section for more details.

To switch 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 CHCFG[ENBL] and CHCFG[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 CHCFG[ENBL] and CHCFG[TRIG] fields are set.

To 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 CHCFG[TRIG] would have no effect due to the assumption that channel 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 Clocking

21.1 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
- Progressive clock switching (PCS)

Note: This chapter covers only clocks that are generated on the device. Protocol clocks that are provided by external devices are covered in the respective chapters where they are used.

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 the “Mode Entry Module (MC_ME)” chapter for details.

The alternative ways to provide the source clock are:

- External oscillator/External crystal
- Internal 16 MHz RC oscillator

From one of these input sources, the internal clocks are generated from PLL0, using PLL0_PHI and PLL0_PHI1 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). Eight Auxiliary Clock Selectors are available so developers can select an independent clock source for each system peripheral. There is one additional clock selector that is used exclusively for the system clock.

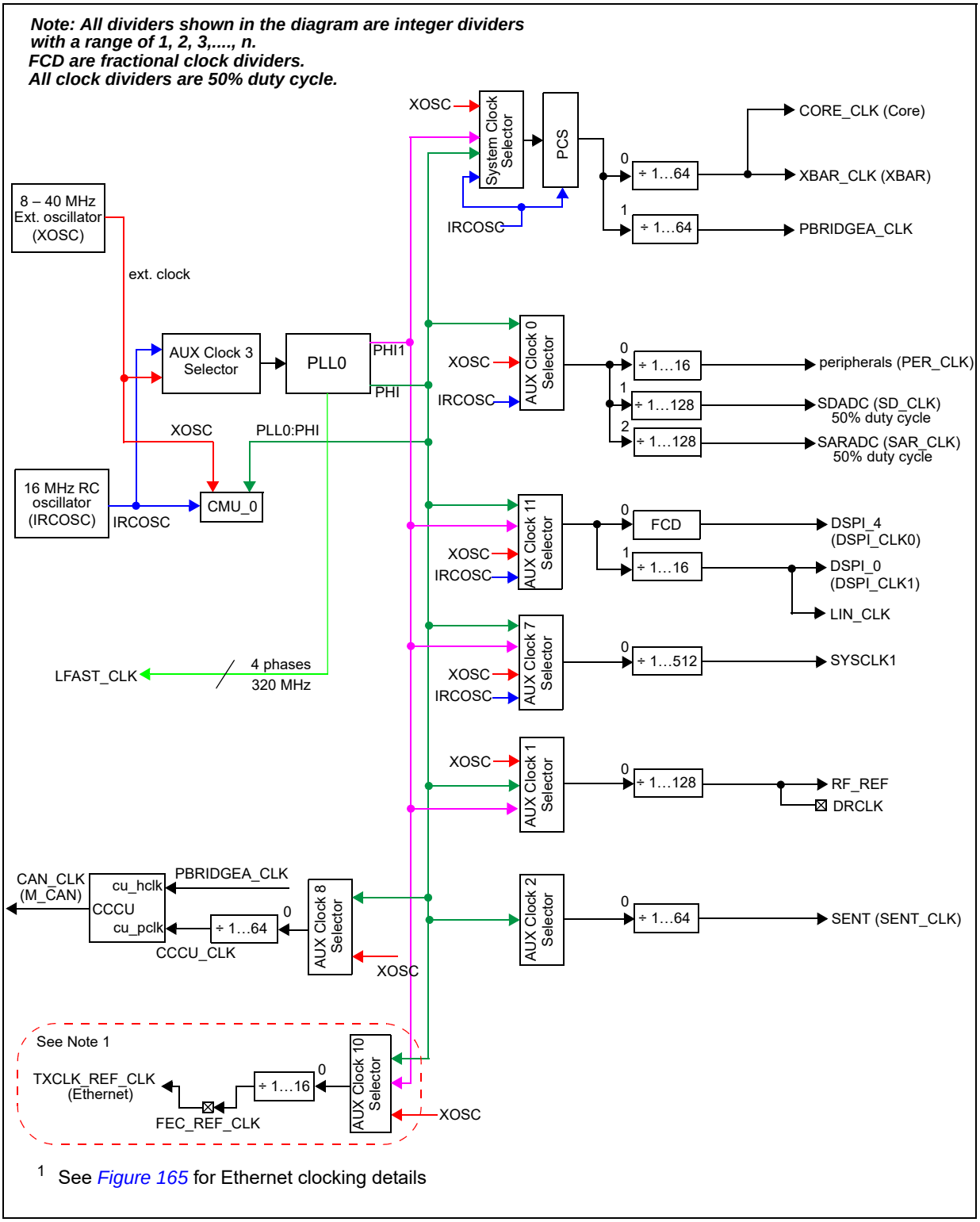
Each of the outputs of the module clock selectors have individual dividers that allow for more clock frequency granularity with the ability to divide the clock selector outputs by up to 512 for a given peripheral. The Clock Monitor Unit (CMU) is connected to IRCOSC, XOSC and PLL0_PHI as shown in *Figure 161: Clock generation* to verify that their clock frequencies stay within necessary operating limits. The action taken when the CMU detects an issue with the monitored clock depends on the CMU configuration. When an issue with a clock is found, the CMU can signal an interrupt or initiate a system reset.

21.2 Clock generation

The top-level clock generation architecture can be seen in *Figure 161*. See the Clock Generation Module (MC_CGM) chapter for details on the Fractional Clock Dividers. 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 161*. All dividers connected to the System Clock Selector must have 50% duty cycle outputs.

Note: The user can select either IRCOSC or XOSC as the reference clock for CMU_0.

Figure 161. Clock generation



1 See *Figure 165* for Ethernet clocking details

21.2.1 MC_CGM registers

Table 207 shows the relationship between the clocks in *Figure 161* and the MC_CGM registers.

Table 207. MC_CGM relationship to clocks

MC_CGM register(s)[1]	Register description	Output clock
CGM_SC_DC0	System Clock Divider Configuration 0	Core (CORE_CLK), XBAR (XBAR_CLK)
CGM_SC_DC1	System Clock Divider Configuration 1	PBRIDGEA_CLK
CGM_AC0_DC0 CGM_AC0_SC	Aux Clock 0 Divider Configuration 0 Aux Clock 0 Select Control	Peripherals (PER_CLK)
CGM_AC0_DC1 CGM_AC0_SC	Aux Clock 0 Divider Configuration 1 Aux Clock 0 Select Control	SDADC (SD_CLK)
CGM_AC0_DC2 CGM_AC0_SC	Aux Clock 0 Divider Configuration 2 Aux Clock 0 Select Control	SARADC (SAR_CLK)
CGM_AC1_DC0 CGM_AC1_SC	Aux Clock 1 Divider Configuration 0 Aux Clock 1 Select Control	RF REF DRCLK
CGM_AC2_DC0	Aux Clock 2 Divider Configuration 0	SENT (SENT_CLK)
CGM_AC3_SC	Aux Clock 3 Select Control	PLL0
CGM_AC7_DC0 CGM_AC7_SC	Aux Clock 7 Divider Configuration 0 Aux Clock 7 Select Control	SYSCLK1
CGM_AC8_DC0 CGM_AC8_SC	Aux Clock 8 Divider Configuration 0 Aux Clock 8 Select Control	M_CAN (CAN_CLK)
CGM_AC10_DC0 CGM_AC10_SC	Aux Clock 10 Divider Configuration 0 Aux Clock 10 Select Control	Ethernet (FEC_REF_CLK)
CGM_AC11_DC0 CGM_AC11_SC	Aux Clock 11 Divider Configuration 0 Aux Clock 11 Select Control	DSPI_4 (DSPI_CLK0)
CGM_AC11_DC1 CGM_AC11_SC	Aux Clock 11 Divider Configuration 1 Aux Clock 11 Select Control	DSPI_0 (DSPI_CLK1), LINFlexD (LIN_CLK)

1. See the Clock Generation Module (MC_CGM) for specifics on register implementation.

21.3 System clock frequency limitations

The maximum frequency of operation for the device system level clocks is given in *Table 208*.

Table 208. Maximum system level clock frequencies

System clock	Max frequency (MHz)[1]
Core	80
XBAR, system RAM	80
PBRIDGE_A	40

Table 208. Maximum system level clock frequencies(Continued)

System clock	Max frequency (MHz)[1]
PBRIDGE_A GTM slot[2]	80
DMA	40
Interrupt controller	40
DSPI_CLK, LIN_CLK	100
Peripheral clock	80
ADCs	14.6
Ethernet FEC_REF_CLK, TXCLK, RXCLK	50, 25, 25

1. The user is responsible to verify that these values are not exceeded.
2. The AIPS bridge (PBRIDGE_A) is capable of clocking the AIPS interface of each peripheral slot individually at the XBAR frequency, or at the XBAR frequency divided by two. The GTM is clocked at the XBAR frequency, and all other peripherals on the AIPS bridge are clocked at the XBAR frequency divided by two.

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 PBRIDGEA_CLK are not required to be at the same frequency. CORE_CLK and XBAR_CLK must always be at the same frequency.

21.3.1 JTAG frequencies

Table 209 shows the maximum frequencies of the JTAG interface.

Table 209. JTAG frequencies

Device	Configuration	e200z2 max. internal JTAG frequency	Max. external JTAG frequency	Notes
PD[1]	Unconfigured clock	—	See electrical characteristics for maximum usable external JTAG frequency	Assumes 16 MHz clock (IRC) with ± 1.5% tolerance, and JTAG usable at 1/2 CPU frequency
	Clock configured to 160 MHz	80 MHz		JTAG usable at 1/2 CPU frequency

1. PD = Production Die

21.4 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.

21.5 Clock sources

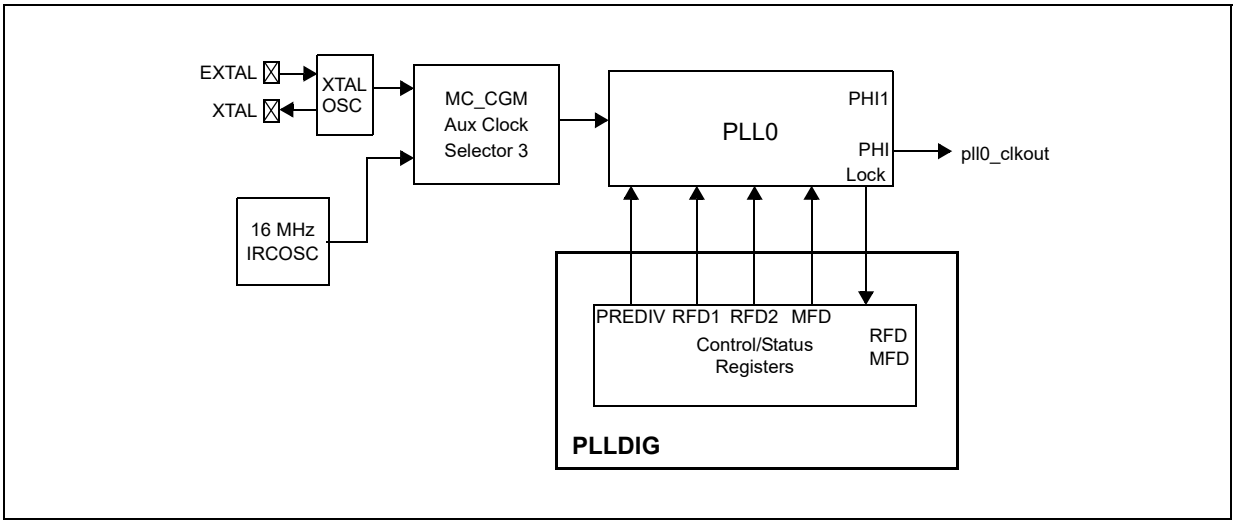
The following clock sources are described in this section:

- PLL0
- External oscillator (XOSC)
- External clock (EXTAL bypass)
- 16 MHz internal RC oscillator (IRCOSC)

21.5.1 PLL

The PLL provides separate system and peripheral clocks as shown in *Figure 161*. The PLL is disabled after power is cycled and must be enabled by software. The block diagram for the PLL is shown in *Figure 162*.

Figure 162. SPC572Lx PLL digital interface block diagram



21.5.1.1 PLL0 (non-FM)

PLL0 provides a non-Frequency Modulated clock to the peripheral IPs.

21.5.1.1.1 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.

21.5.1.1.2 PLL0 output clocks

The output clocks from PLL0 are PHI and PHI1. The PHI output clock drives the peripheral clock and the system clock.

21.5.1.1.3 PLL register write protection

The PLL registers with write protection are defined in *Table 210*. 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. The REG_PROT module is used to implement the PLL register write protection.

Table 210. PLL register write protection

Offset	Register[1]	Protection
00h	PLL0 Control Register (PLL0CR)	Yes
04h	PLL0 Status Register (PLL0SR)	No
08h	PLL0 Divider Register (PLL0DIVR)	Yes

1. See Register Protection Configuration chapter for bit field details.

21.5.1.2 PLL register reset values

The reset values for the device-specific PLLDIG registers are shown in *Table 211*.

Table 211. PLL register reset values

Offset (hex)	Register	Reset value (hex)
0008	PLL0DV — PLL0 Divider Register	0000_0000

21.5.1.3 Loss of clock detection

Loss of clock detection should be enabled at all times when the PLL is 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.

21.5.1.4 PLLDIG initialization information

Configure PLL and related modules.

1. With PLL disabled, program PLL clock source in MC_CGM_AC3_SC[SELCTL]. Default source is 16 MHz IRCOSC. Write MC_CGM_AC3_SC[SELCTL] = 1 to select XOSC as source.
2. Program PLLDV[PREDIV], PLLDV[RFDPHI1], PLLDV[MFD] and PLLDV[RFDPHI].
3. If desired, modify the XOSC_CTL[EOCV] to adjust the external oscillator stabilization count, used when the external oscillator is turned on (by the MC_ME).
4. If necessary, modify the CMU frequency meter values (CMU_MDR and CMU_FDR) used to monitor the XOSC frequency. XOSC frequency is compared to the IRCOSC source when XOSC is turned on.

Turn on XOSC and PLL.

5. Configure a mode configuration for turning on PLL and XOSC. In a mode configuration register (for example, MC_ME_DRUN_MC) set MC_ME_<mode>_MC[XOSCON] = 1 and MC_ME_<mode>_MC[PLLON] = 1. If desired, also set MC_ME_<mode>_MC[SYSCLK] = 2 for this new mode configuration to use PLL PHI output) as the sysclk.
6. Enter that mode by two writes to MC_ME_CTRL register to enter that mode. This is required even if entering the same mode.

Wait for the mode transition to complete.

7. Wait for the mode transition to complete by polling MC_ME_GS[S_MTRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog,

should be used to make sure the mode transition completes. Mode transition will NOT complete until:

– XOSC counter expires (if the new mode configuration changes MC_ME_<mode>_MC[XOSCON] = 1), and

– PLL is locked (if the new mode configuration changes MC_ME_<mode>_MC[PLLON] = 1)

8. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

Note: See the "Mode Entry Module (MC_ME)" chapter in this Reference Manual more details.

21.5.2 External oscillator (XOSC)

The external oscillator (XOSC) is provided as a reference for the on-chip PLL. It can also can be used as a clock source to the ADCs, timer, serial interfaces, LFAST, and as a system clock source. It is available for observation on the CLKOUT pin, 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
- Reference clock to CMU0 (IRCOSC trimming)
- Option to drive the CAN protocol clock directly from the XOSC

The XOSC provides support for 8 – 44 MHz crystal inputs, has integrated load capacitors and ALC to remove the need for external series resistor.

21.5.2.1 XOSC Start-up

Enabling of the oscillator at startup is determined by a bit in the UTEST row of flash memory (UTEST DCF Miscellaneous row, bit 7). The default value has the XOSC disabled (UTEST Misc bit 7 = 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 6 = 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 5:1 = 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_40MHz 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 10 = 0).

See the flash memory chapter for more information on the UTEST row programming.

21.5.2.2 XOSC register write protection

The XOSC registers with write protection are defined in *Table 212*. 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. The REG_PROT module is used to implement the XOSC register write protection.

Table 212. XOSC register write protection

Offset	Register(1)	Protections?
00h	XOSC_CTL—XOSC Control Register	Yes

1. See Register Protection Configuration chapter for bit field details

21.5.2.3 XOSC reset value

Table 213 shows the default reset value for the XOSC registers.

Table 213. XOSC register reset values

Offset	Register	Reset value
00h	XOSC_CTL—XOSC Control Register	8030_8000h

21.5.3 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 PLL. 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.

21.5.3.1 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. Other registers allow reading of the settings of IRCOSC temperature sensor, voltage regulator and capacitor trimming values that are set by the factory. The IRCOSC is always enabled when the device is powered.

21.5.3.2 IRCOSC reset value

Table 214 shows the default reset value for the IRCOSC registers. The values of the IRCOSC registers are loaded from TEST DCF flash memory (16 MHz RCOSC_1, 16 MHz RCOSC_2, 16 MHz RCOSC_3, 16 MHz RCOSC_4) (see Device Configuration Format Records chapter for details).

Table 214. IRCOSC register reset values

Offset	Register	Reset value
04h	NT—IRCOSC Native Trimming Register	0000_0000h
08h	TT—IRCOSC Temperature Trimming Register	0000_0000h

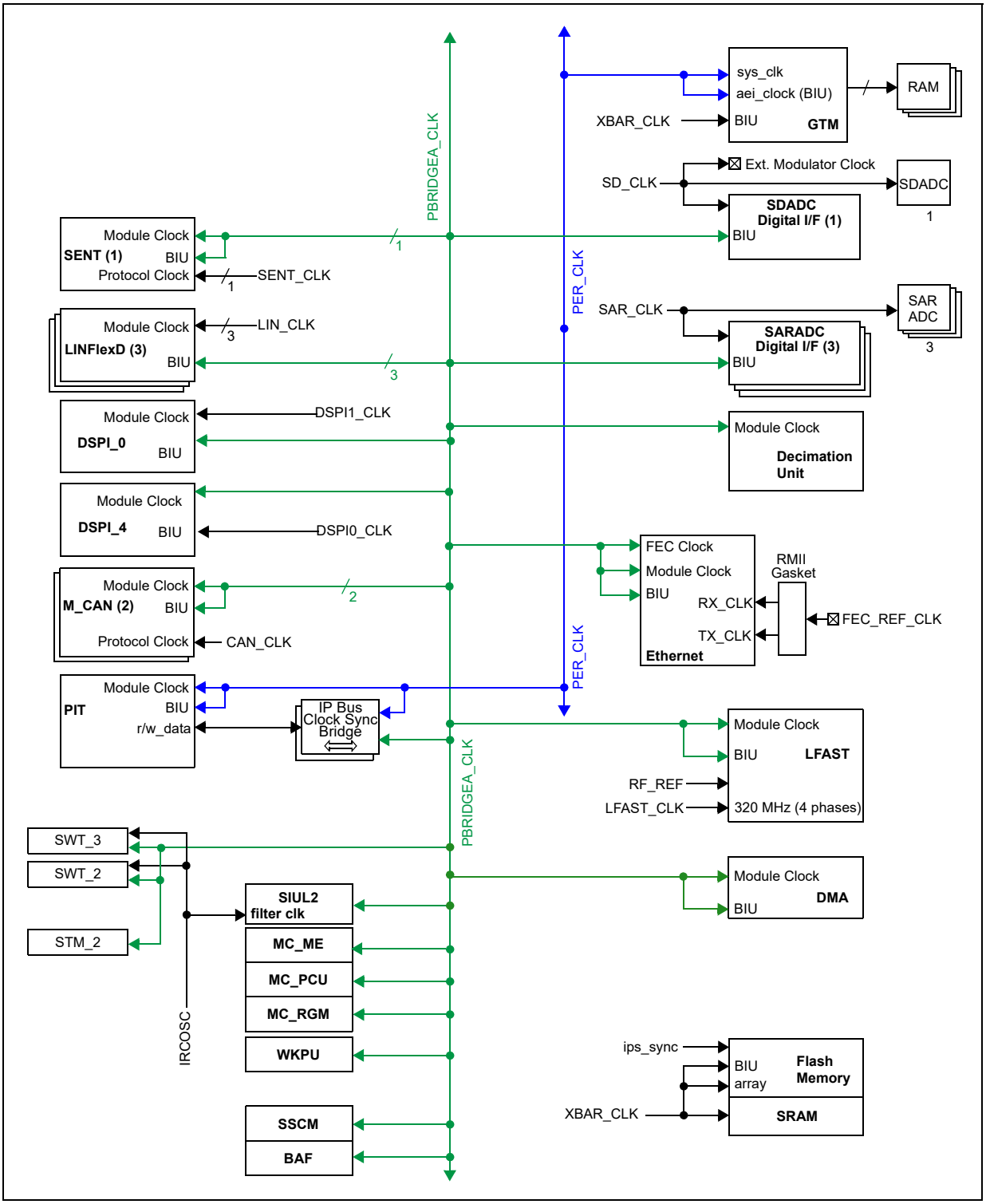
21.5.3.3 IRCOSC register write protection

The IRCOSC registers do not utilize write protection.

21.6 Peripheral clocks

Figure 163 shows the clock distribution to the cores and peripheral modules.

Figure 163. Clock distribution



21.6.1 LFAST clocking

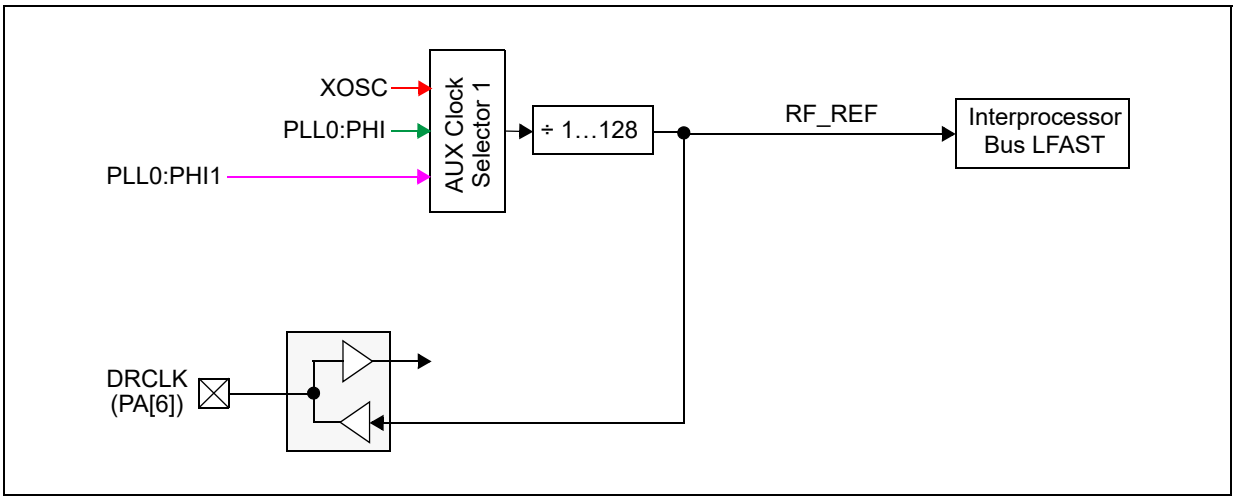
The LFAST is clocked with a 320 MHz clock signal (4 phases) coming from PLL0.

If SPC572Lx is operated as LFAST slave the reference clock coming from the LFAST master is fed as an external clock signal (EXTAL bypass) to be used by the PLL0 as clock input.

If SPC572Lx is operated as LFAST master the reference clock is selectable by “AUX Clock Selector 1” and provided to the external LFAST slave by the DRCLK pin.

Figure 164 shows the clock routing for the LFAST module on the device, including connections to the DRCLK pin.

Figure 164. Device LFAST clocking



21.6.1.1 LFAST and Ethernet use cases

On SPC572Lx, LFAST and Ethernet are not used simultaneously and this leads to two different use-cases for the clock tree configuration, as shown in *Table 215*.

Table 215. LFAST and Ethernet use cases

Use case	PLL0 f_{VCO}	PHI	PHI1	XBAR_CLK	PBRIDGEA_CLK	PER_CLK	DSPI1_CLK LIN_CLK	LFAST_CLK	FEC_REF_CLK
1(1)	800 MHz	80 MHz	100 MHz	80 MHz	40 MHz	80 MHz	80 MHz or 100 Hz	not running	50 MHz
2(2)	1280 MHz	80 MHz	—	80 MHz	40 MHz	80 MHz	80 MHz	320 MHz (4 phases)	not running

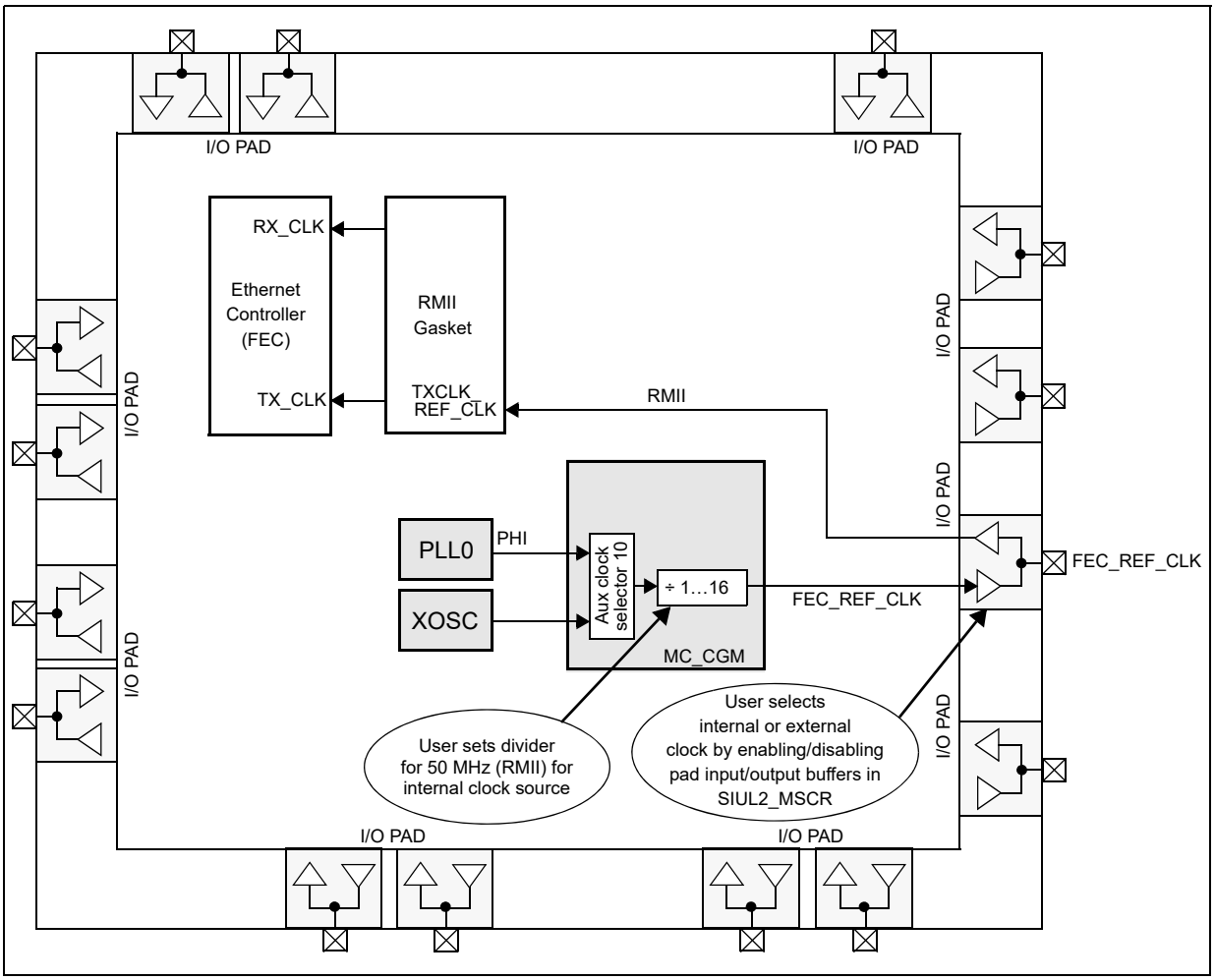
1. In this use case the maximum achievable baud rate for the LINFlexD is 25 MHz, if LIN_CLK is configured for 100 MHz.
2. In this use case the maximum achievable baud rate for the LINFlexD is 20 MHz.

21.6.2 Ethernet clocking

Both internal and external clocks sources for RMII Ethernet interface mode are supported. The RMII clock (FEC_REF_CLK) is 50 MHz. *Figure 165* gives the connections for the Ethernet clocks.

The FEC_REF_CLK can be generated internally or input to the device. FEC_REF_CLK can be generated internally and fed to both the PHY device and to the RMII gasket by enabling the FEC_REF_CLK pad input and output buffers. FEC_REF_CLK can also be provided externally as an input to the device.

Figure 165. Device Ethernet clocking



21.6.3 M_CAN clocking

The M_CAN modules are clocked with a protocol clock CAN_CLK coming from the Clock Calibration Unit on CAN (CCCU). The CCCU is clocked by CCCU_CLK.

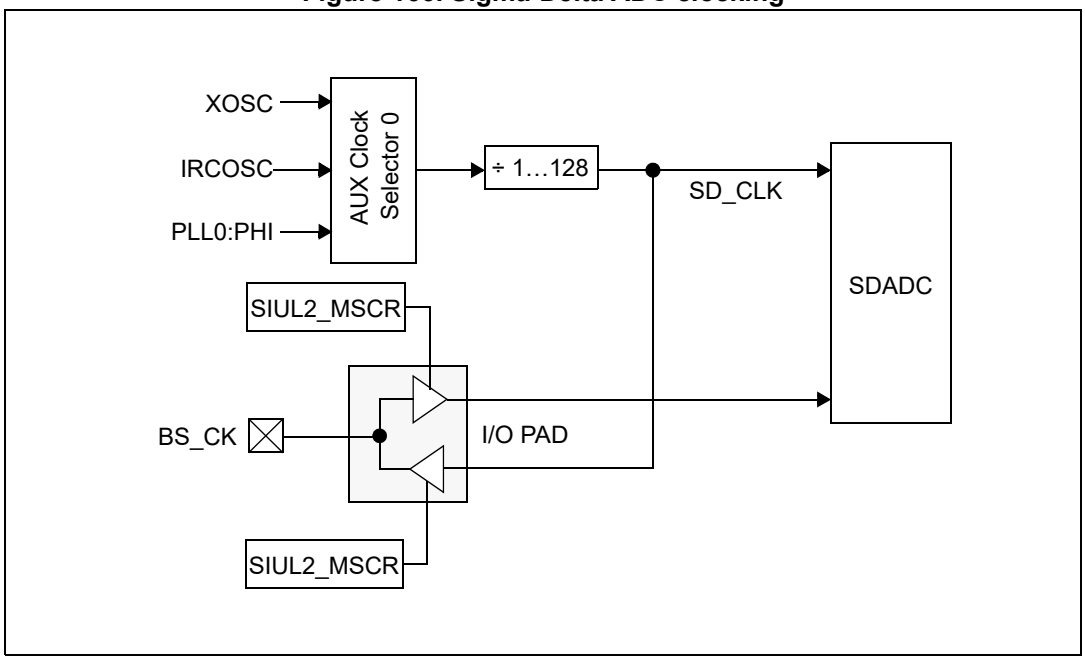
The CCCU generates a calibrated CAN clock from an on-chip unprecise clock (RC oscillator) by evaluating CAN messages received by the attached M_CAN_1. For more details refer to the section *Section 44.3.16: Clock Calibration on M_CAN*.

21.6.4 Sigma-Delta ADC clocking

The analog portion of the Sigma-Delta ADC (SDADC) works with either an internal or external modulator. The internal clock (SD_CLK, see *Figure 163*) is always used when the internal modulator in the SDADC is enabled. This device supports either an external input clock (BS_CK), or internally generated output clock (SD_CLK) when using an external modulator. The output clock is generated internally and connected to the external modulator clock package pin (BS_CK). There is a dedicated external modulator clock pin for each SDADC.

The SDADC contains an internal mux for selection between the internal and external clocks. The internal clock is selected when the internal modulator is enabled, and the external clock is selected when it is disabled. In order to provide an internal clock option when the internal modulator is enabled, the SD_CLK output from the MC_CGM must be connected to the BS_CK pin as shown in *Figure 166*. Control of the input/output direction of the BS_CK pin is handled by writing the MSCR in the SIUL2. SDADC is connected as shown in *Figure 166*.

Figure 166. Sigma-Delta ADC clocking



21.6.5 Successive Approximation Register ADC clocking

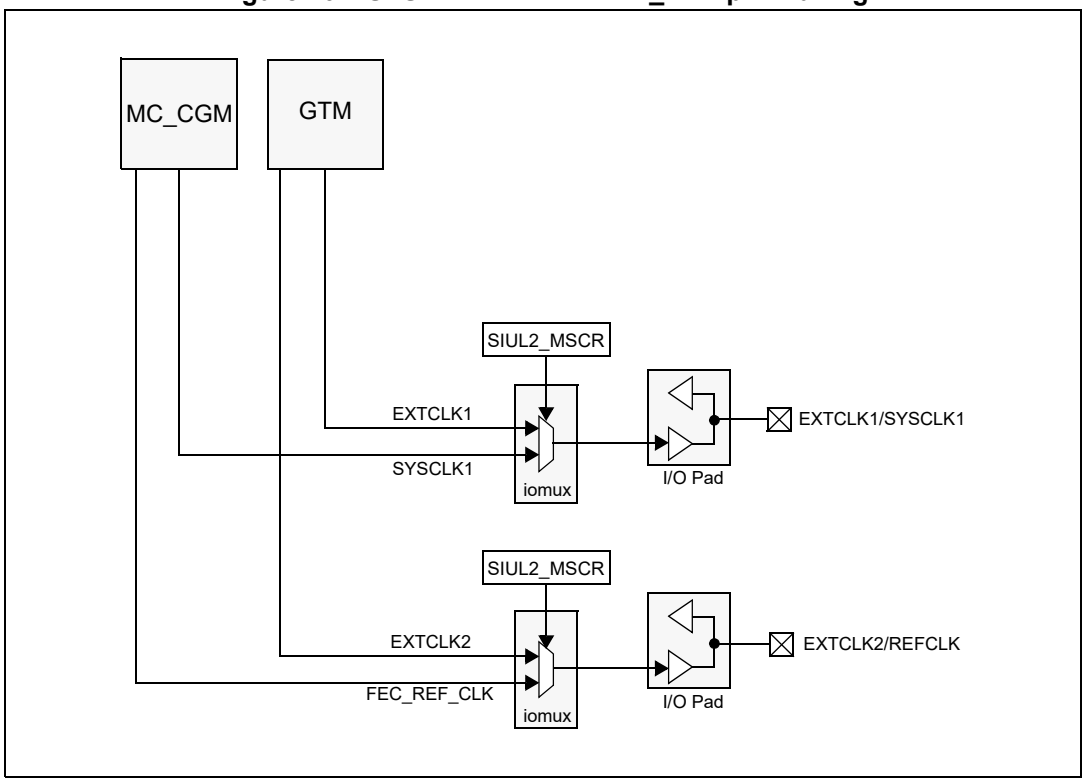
The analog portion of the Successive Approximation Register ADC (SARADC) requires a dedicated clock SAR_CLK, which should be configured, complying with the frequency constraints defined into the device datasheet. SAR_CLK must be less than PBRIDGEx_CLK.

21.6.6 System Clock/GTM output clock/Ethernet clock pin muxing

There is one system clock output pin on the device, SYSCLK[1]. The output clock from either of the PLLs, the internal oscillator (IRCOSC), or external oscillator (XOSC) can be individually selected for SYSCLK[1]. See *Figure 161*.

There are two GTM output clocks available on package pins, EXTCLK[2:1]. EXTCLK[1] is muxed on the same pins as SYSCLK[1]. EXTCLK[2] is muxed on the same pin as the Ethernet RMII clock (FEC_REF_CLK). The selection of EXTCLK[*n*], SYSCLK[*n*] or FEC_REF_CLK is done in the SIUL2_MSCR registers and shown in *Figure 167*.

Figure 167. SYSCLK/EXTCLK/REF_CLK pin muxing



21.7 Clock monitoring

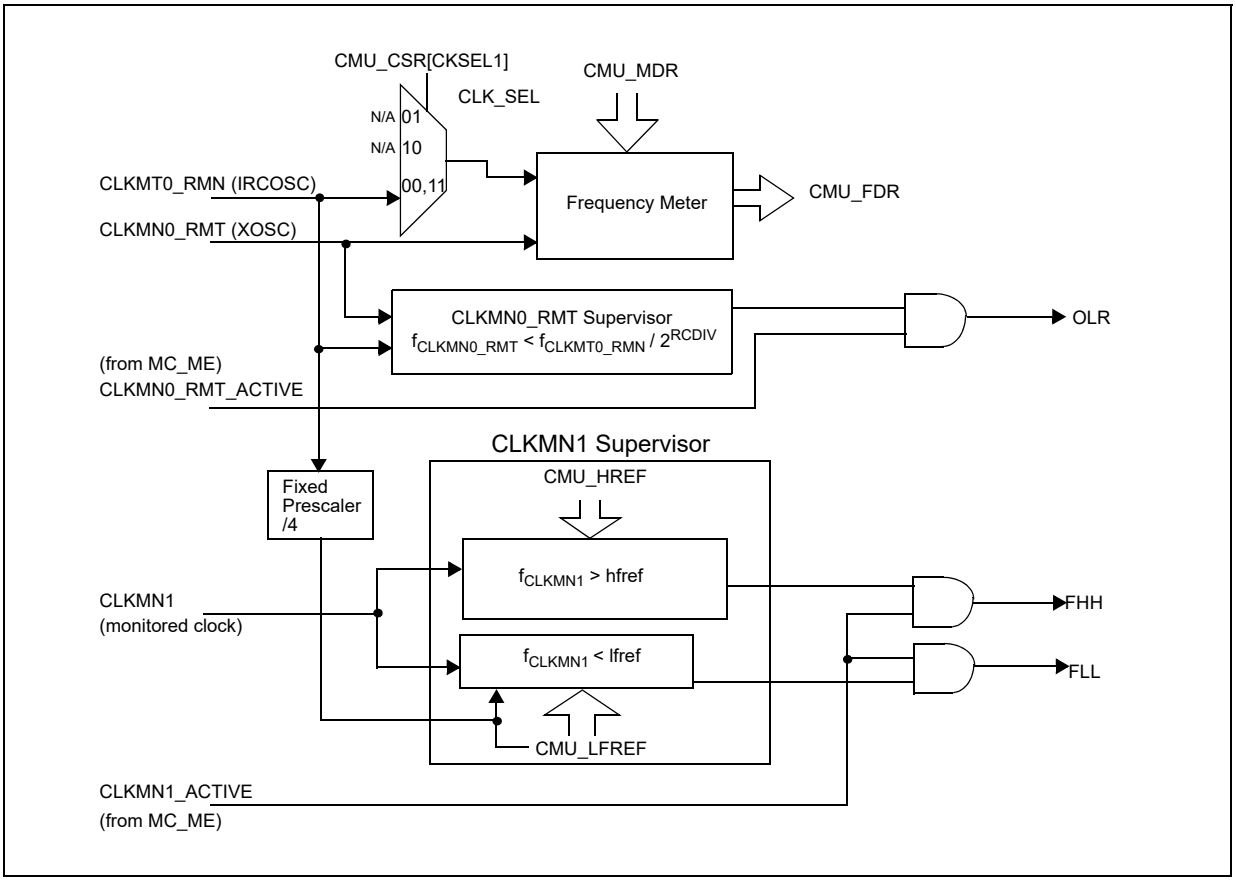
For all safety relevant clocks the microcontroller uses a clock monitoring unit (CMU) to detect a missing clock or incorrect frequency. The IRCOSC and XOSC are used as the clock monitor references. Detailed information on the CMU can be found in the Clock Monitor Unit chapter.

21.7.1 CMU configuration

This section explains the CMU configuration.

Figure 168 shows the block diagram for CMU0 on the SPC572Lx.

Figure 168. CMU0 block diagram



21.7.1.1 Clock input sources

Table 216 shows the clocks that are monitored by the 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 the clock monitor. Only CMU0 implements the XOSC monitor. CMU0 uses the IRCOSC clock to measure if the XOSC is too low. CMU0 can also be used to calibrate the IRCOSC frequency using the XOSC.

Table 216. Clock input sources

Clock module	Monitored clock
CMU0	PLL0:PHI, XOSC, IRCOSC

21.7.1.2 CMU registers availability

Table 217 specifies the CMU registers available on this device.

Table 217. CMU registers availability

Address offset	Register	CMU
0000h	CMU_CSR	CMU0
0004h	CMU_FDR	CMU0
0008h	CMU_HFREFR	CMU0
000Ch	CMU_LFREFR	CMU0
0010h	CMU_ISR	CMU0
0014h	Reserved	—
0018h	CMU_MDR	CMU0

21.7.1.3 CMU register write protection

The CMU registers have write protection as shown in *Table 218*. A soft lock can be unlocked by software after being previously locked. A hard lock is only unlocked by a reset once locked. The REG_PROT module is used to implement the CMU register write protection.

Table 218. CMU register write protection

Offset	Register[1]	Protections?
0000h	CMU Control Status Register (CMU_CSR)	Yes
0004h	CMU Frequency Display Register (CMU_FDR)	No
0008h	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	No
000Ch	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	No
0010h	CMU Interrupt Status Register (CMU_ISR)	No
0014h	Reserved	No
0018h	CMU Measurement Duration Register (CMU_MDR)	No

1. See Register Protection chapter for bit field details.

21.7.2 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.

21.7.3 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.

21.7.4 Internal RC oscillator (IRCOSC) monitor

The period of the IRCOSC can be measured in CMU0, using the XOSC as a reference. This allows for application trimming of the IRCOSC frequency.

21.8 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 169* gives a high-level view of the loss of clock logic.

21.8.1 Loss of PLL/XOSC clock

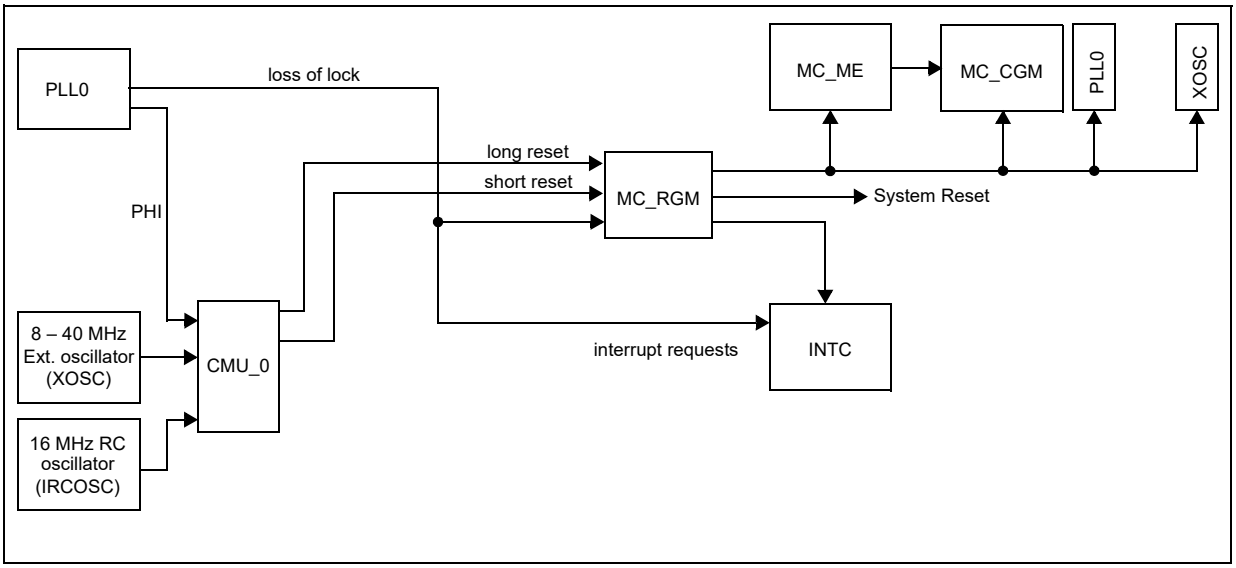
As shown in *Figure 169*, each loss of lock signal from each PLL and the XOSC failure signals trigger an interrupt. In the case of an XOSC failure, only the backup clock in the PLL can provide clocks to the system in order to perform a system clock switch. There is no automatic system clock switch, and 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 also in this situation the PLL backup clock is the only available clock to clock the system through an MC_ME mode switch.

21.8.2 Loss of IRCOSC clock

The frequency of the IRCOSC clock is monitored by the frequency meter in CMU_0. Since the IRCOSC is the boot and backup clock, a failure is a catastrophic failure and generates a system reset.

Figure 169. Loss of clock logic



21.9 Progressive clock switching

The Progressive Clock Switching (PCS) block is shown in the system-level clock diagram in *Figure 161*. It is used for ramping the system clocks. See the clock Generation Module (MC_CGM) chapter for more information on the PCS.

22 PLL Digital Interface (PLLDIG)

22.1 Introduction

The MCU provides a user interface and control over the PLL system (PLLDIG).

22.2 Block Diagram

The block diagram of the PLLDIG block is found in the “Clocking chapter” of the *SPC572Lx Reference Manual*.

22.3 Features

The PLLDIG has the following features:

- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for reducing the output clock frequency without causing the PLL to lose lock.
- Lock detect circuitry reports when the PLL has achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.
  - User-selectable ability to generate an interrupt request upon loss of lock.
  - User-selectable ability to generate a system reset upon loss of lock.

22.4 Modes of operation

The operating mode of the PLLs is determined by the value of PLL*n*CR[CLKCFG]. The mode of operation for the PLL is:

- Normal mode with reference, and PLL 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.

22.4.1 Normal mode with reference, PLL enabled

In the normal mode, the PLL receives an input clock from the reference and the pre-divider. The PLL multiplies the frequency to create the PLL 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.

The PLL generates a non-modulated clock which drives PLL_PHI1.

22.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs.

22.5.1 Memory map

Table 219 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 219. PLL Digital Interface memory map

Offset (hex)	Register	Access	Reset Value (hex)	Section/Page
0000	PLLCR — PLL Control Register	R/W	0000_0000h	*on page 430*
0004	PLLSR — PLL Status Register	R/W	0000_0000h	*on page 431*
0008	PLLDV — PLL Divider Register	R/W	zzzz_z0zzh (1)	*on page 432*
000C–0012	Reserved	—	—	—
0014–001C	Reserved	—	—	—
0018	Reserved	—	—	—
0020–0030	Reserved	—	—	—
0034–0038	Reserved	—	—	—

1. See the “Clocking” chapter for default reset values.

22.5.2 Register descriptions

Some of the register reset values are specifically configured for each unique device by external configuration signals or parameters.

22.5.2.1 PLL Control Register (PLLCR)

Offset: 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	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	Reserved(1)	CLKCFG(2)		EXPDIE	—(1)	—(1)	—(1)	LOLIE	LOLRE	—(1)	—(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be written at any time, but writes are ignored. Reads returns previously written value.
2. CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_<mode>_MC register (see the “Mode Entry Module (MC_ME)” chapter and Clock configuration for details).

Figure 170. PLL Control Register (PLLCR)

Table 220. PLLCR field descriptions

Field	Description
0:21	Reserved
22:23 CLKCFG	Clock Configuration. This field indicates the operational state of the PLL. 00PLL off. 01Reserved 10Reserved. 11Normal mode with PLL running When PLLs are externally powered down the CLKCFG field changes to 00b. When external power down is removed, these bits will read 11b.
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
25	Reserved
26	Reserved
27	Reserved
28 LOLIE	Loss of lock interrupt enable. The LOLIE bit enables a loss of lock interrupt request when PLLSR[LOLF] = 1. If PLLSR[LOLF] = 0 or PLLCR[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.

Table 220. PLLCR field descriptions(Continued)

Field	Description
29 LOLRE	Loss of lock reset enable. The PLLCR[LOLRE] bit determines whether system reset is asserted or not upon a loss of lock indication from PLL. If both PLL0CR[LOLRE] = 1 and PLL0CR[LOLIE] = 1, PLL0CR[LOLRE] has precedence and the device asserts reset with the interrupt not being taken. 0 Ignore loss of lock. Reset not asserted. 1 Assert reset on loss of lock when operating in normal mode.
30	Reserved
31	Reserved

22.5.2.2 PLL Status Register (PLLSR)

Offset: 0004h Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	EXTPDF	0	0	0	LOLF	LOCK	0	0
W									w1c		*(1)		w1c		*(1)	
Reset	0	0	0	0	0	0	0	0	0	—	0	—	0	0	0	0

Figure 171. PLL Status Register (PLLSR)

1. This field is reserved, read only.

Table 221. PLLSR field descriptions

Field	Description
0:23	Reserved
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
25	Reserved
26	Reserved
27	Reserved
28 LOLF	Loss of lock flag. This bit provides the interrupt flag for the loss of lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. The flag is set when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.

Table 221. PLLSR field descriptions(Continued)

Field	Description
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. 0 PLL is unlocked. 1 PLL is locked.
30	Reserved
31	Reserved

22.5.2.3 PLL Divider Register (PLLDV)

The PLLDV register provides the PHI/PHI1 output clock reduced frequency dividers, pre-divider, and loop divider. The values of PREDIV and MFD should not be changed when the PLL is on and locked (working in Normal mode). If these fields are changed without powering down the PLL, the PLL will lose lock and generate either a reset or an interrupt, based on which is enabled. The reduced frequency divider fields can be modified at anytime, but the changes only become effective after PLL is disabled, then re-enabled.

Offset: 0008h 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)

Figure 172. PLL Divider Register (PLLDV)

1. See the Clocking chapter for reset value information.

Table 222. PLLDV field descriptions

Field	Description
0	Reserved
1:4 RFDPHI1	PHI1 reduced frequency divider. This 4-bit field determines the VCO clock post divider for driving the PHI1 output clock. 00xx Reserved 0100 Divide by 4 0101 Divide by 5 .... 1110 Divide by 14 1111 Divide by 15
5:9	Reserved

Table 222. PLLDV field descriptions(Continued)

Field	Description
10:15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. 00 Reserved 01 Divide by 1 02 Divide by 2 03 Divide by 3 .... 3E Divide by 62 3F Divide by 63
16	Reserved
17:19 PREDIV	Input clock pre-divider. This 3-bit field controls the value of the divider on the input clock. The output of the pre-divider circuit generates the reference clock to the PLL analog loop. The PREDIV value 000b causes the input clock to be inhibited. 000 Clock inhibit 001 Divide by 1 010 Divide by 2 .... 110 Divide by 6 111 Divide by 7
20:24	Reserved
25:31 MFD	Loop multiplication factor divider. This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 00 Reserved .... 07 Reserved 08 Divide by 8 .... 7E Divide by 126 7F Divide by 127

22.5.3 Register classification for safety requirements

This module is classified as a safety Vital Module (ViMo). Hence, the registers of this module have been classified as shown in the SPC572Lx Reference Manual “Functional Safety” chapter.

22.6 Functional description

This section explains the operation and configuration of the PLLDIG module.

22.6.1 Input clock frequency

The PLL is designed to operate over an input clock frequency range. The operating ranges for each PLL are discussed in detail in the *SPC572Lx Data Sheet*.

22.6.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLLDV register, and calculated according to the following equations:

Equation 4

$$f_{pll_phi} = f_{pll_ref} \times \frac{PLLDV[MFD]}{PLLDV[PREDIV] \times PLLDV[RFDPHI]}$$

Equation 5

$$f_{pll_phi1} = f_{pll_ref} \times \frac{PLLDV[MFD]}{PLLDV[PREDIV] \times PLLDV[RFDPHI1]}$$

The relationship between the VCO frequency f_{VCO} and the output frequency of the PLL is determined by the PLLDV register, according to the following equations:

Equation 6

$$f_{pll_VCO} = f_{pll_ref} \times \frac{2xPLLDV[MFD]}{PLLDV[PREDIV]}$$

When programming the PLL, user software must not violate the maximum system clock frequency or max/min VCO frequency specification. Furthermore, the PLLDV[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 PLLSR[LOCK] flag are immediately negated if the fields of PLLDV are changed without powering down the analog PLLs. However, changing the PLLDV field without powering down the PLL is not recommended. If the updated values are able to maintain the PLL frequency within operating parameters, no loss of lock is generated.

When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLLSR[LOCK] flag is held negated. After the timer expires, the PLLSR[LOCK] flag reflects the value coming from the PLL lock detection circuitry. To prevent an immediate reset, the PLLCR[LOLRE] bit of the respective PLLs must be cleared before doing any of the above operations.

Note: The PLL must be powered down and powered up by configuring, then executing a mode change in the MC_ME_<mode>_MC before PLLCR[PREDIV], PLLCR[MFD] or the input reference clock values are modified.

The recommended procedure to program the PLLs and engage normal mode is shown in *Section 22.7: Initialization information*.

22.6.3 Loss of lock

The PLL digital interface registers provide the flexibility to select whether to generate an interrupt assert system reset or do nothing in the event that the PLL loses lock according to the available PLL*n*CR options.

Loss of lock reset and interrupt are only generated when the PLL is operating in normal mode.

The lock indication from the analog PLL is synchronized and stored in the status register. When the analog PLL loses lock or regains lock, it will be immediately indicated in the status register (after synchronization time).

A pair of counters monitor the reference and feedback clocks to determine when the system has acquired frequency lock. Once the PLL has locked, the counters continue to monitor the reference and feedback clocks and report whether the PLL has lost lock. The lock status is indicated in the status register.

1. Power down by writing MC_ME_<*mode*>_MC[PLL0ON] = 0, followed by a mode change
2. Write a mode change to MC_ME_<*mode*>_MC[PLL0ON].
3. Power up the PLL by writing MC_ME_<*mode*>_MC[PLL0ON] = 1, followed by a mode change.
4. Write a mode change to MC_ME_<*mode*>_MC[PLL0ON].

22.7 Initialization information

Coming out of reset the PLL is disabled per the DRUN mode configuration register, MC_ME_DRUN_MC. The PLLDIG initialization procedure is described in the “Clocking” chapter.

23 Clock Monitor Unit (CMU)

23.1 Introduction

The Clock Monitor Unit (CMU), also referred to as Clock Quality Checker or Clock Fault Detector, serves three purposes:

- Measures the frequency of clock sources CLKMT0_RMN with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and also detects if the monitored clock frequency leaves an upper or lower frequency boundary

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the division factor given by the RCDIV field of CMU Control Status Register and reports this event. The CMU can also monitor CLKMN1 and generate events when the CLKMN1 frequency is less than the CLKMT0_RMN frequency divided by four or if CLKMN1 leaves an upper or lower frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of one clock source against a reference clock. This is useful to allow the calibration of the metered clocks (such as CLKMT0_RMN), as well as to be able to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

Note: See the “Clocking” chapter for the number of instances of the CMU in this device.

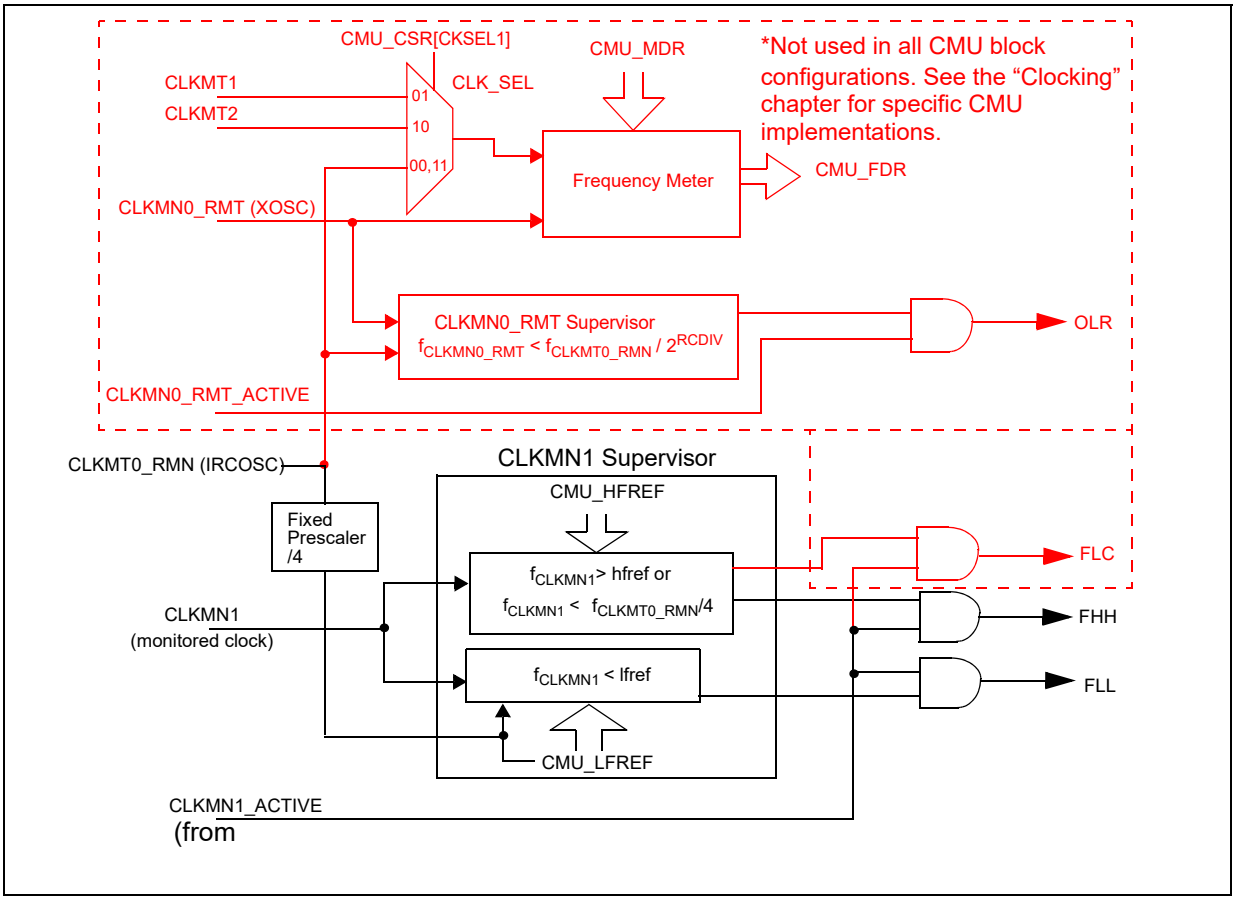
23.1.1 Main features

- CLKMT0_RMN frequency measurement with CLKMN0_RMT as the reference clock.
- CLKMN0_RMT monitoring with respect to CLKMT0_RMN/n clock.
- CLKMN1 frequency monitoring with respect to CLKMT0_RMN / 4 and upper or lower frequency boundary.
- Event generation for various failures detected inside monitoring unit.

23.2 Block diagram

The full block diagram of the CMU is shown in *Figure 173*. See the “Clocking” chapter for device specific use of each CMU module.

Figure 173. Clock Monitor Unit diagram



23.3 Signals

Table 223 describes the signals on the boundary of the CMU (in alphabetical order).

Table 223. Signal description

Signal	I/O	Description
CLKMN0_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference — Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.
CLKMN1	I	Monitored Clock Signal 1 — Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference — Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.

Note: See the "Clocking" chapter for device specific clock sources of each CMU.

23.4 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 224*.

Table 224. CMU memory map

Offset (hex)	Register	Location
0000	CMU Control Status Register (CMU_CSR)	*on page 438*
0004	CMU Frequency Display Register (CMU_FDR)	*on page 439*
0008	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	*on page 440*
000C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	*on page 440*
0010	CMU Interrupt Status Register (CMU_ISR)	*on page 441*
0014	Reserved	—
0018	CMU Measurement Duration Register (CMU_MDR)	*on page 442*

Note: See the "Clocking" chapter for register and field availability details.

23.4.1 Register descriptions

23.4.1.1 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 utilize this feature. See the "Clocking" chapter for device specific CMU implementation details.

Figure 174. CMU Control Status Register (CMU_CSR)

Table 225. CMU_CSR field descriptions

Field	Description
0:7	Reserved
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. 0 Frequency measurement is completed or not yet started 1 Frequency measurement is not completed
9:21	Reserved
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. 00 CLKMT0_RMN is selected 01 Reserved 10 Reserved 11 CLKMT0_RMN is selected
24:28	Reserved
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

23.4.1.2 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 226. CMU_FDR field descriptions

Field	Description
0:11	Reserved
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: $f_{sel} = (f_{CLKMN0_RMT} \times CMU_MDR[MD]) / CMU_FDR[FD]$

23.4.1.3 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 227* show 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 227. CMU_HFREFR field descriptions

Field	Description
0:19	Reserved
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).

23.4.1.4 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 228* show 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 228. CMU_LFREFR field descriptions

Field	Description
0:19	Reserved
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).

23.4.1.5 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	0	FHHI	FLLI	OLRI(1)
W													—(2)	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details.

2 Field is read only.

Figure 178. CMU Interrupt Status Register (CMU_ISR)

Table 229. CMU_ISR field descriptions

Field	Description
0:28	Reserved
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 **Note:** 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.

Note: Before entering low-power stop modes, all clock frequency measurements in CMU should be disabled. Failing to do so may result in spurious interrupts.

23.4.1.6 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 230. CMU_MDR field descriptions

Field	Description
0:11	Reserved
12:31 MD	Measurement duration bits. This field displays the measurement duration in terms of selected clock (CLKMT0_RMN) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR [SFM] = 1.

23.5 Functional description

This section describes the functionality of the CMU.

23.5.1 Frequency meter

The purpose of frequency meter is to evaluate the deviation from the nominal metered source (such as, CLKMT0_RMN) frequencies. This in turn allows either recalibration of these clocks or other timing corrections. Programming of the CMU_CSR[CKSEL1] field is used to select one of metered clocks from a multiplexer that drives a simple Frequency Meter (see *Figure 173*). 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 7 $\mathbf{f_{sel} = (f_{CLKMN0_RMT} \times MDR[MD]) / FDR[FD]}$

23.5.2 CLKMN0_RMT supervisor

If frequency of CLKMN0_RMT is smaller than frequency of CLKMT0_RMN / $2^{CMU_CSR[RCDIV]}$ and CLKMN0_RMT is ‘ON’ as signaled by the MC_ME, then:

- 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.

23.5.3 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.

If f_{CLKMN1} is greater than the reference value determined by fields CMU_HFREFR[HFREF] and CLKMN1 is ‘ON’ as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FHHI]
- The CMU asserts the FHH signal

Note: An example of determining the $HFREF_{Actual}$ value is as follows. Assume a f_{CLKMT0_RMN} = 16 MHz with an accuracy of +/-5%. In order to monitor f_{CLKMN1} = 200 MHz, the ideal $HFREF_{Ideal}$ = 800. The actual HFREF value will be 842 when the accuracy is taken into consideration ($HFREF_{Actual}$ = ($HFREF_{Ideal}$ + 2) × 1.05).

If f_{CLKMN1} is less than a reference value determined by the bits CMU_LFREFR[LFREF] and the CLKMN1 is 'ON' as signaled by the MC_ME, then:

- 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*

Note: For example, assuming f_{CLKMT0_RMN} is 16MHz with +/-5% variation, in order to monitor the f_{CLKMN1} (200MHz), the ideal $LFREF_{Ideal}$ is 800. The adjusted value considering the tolerances will be 758 ($LFREF_{Actual}$ = ($LFREF_{Ideal}$ -2) x 0.95).

Note: 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.

24 Clock Generation Module (MC_CGM)

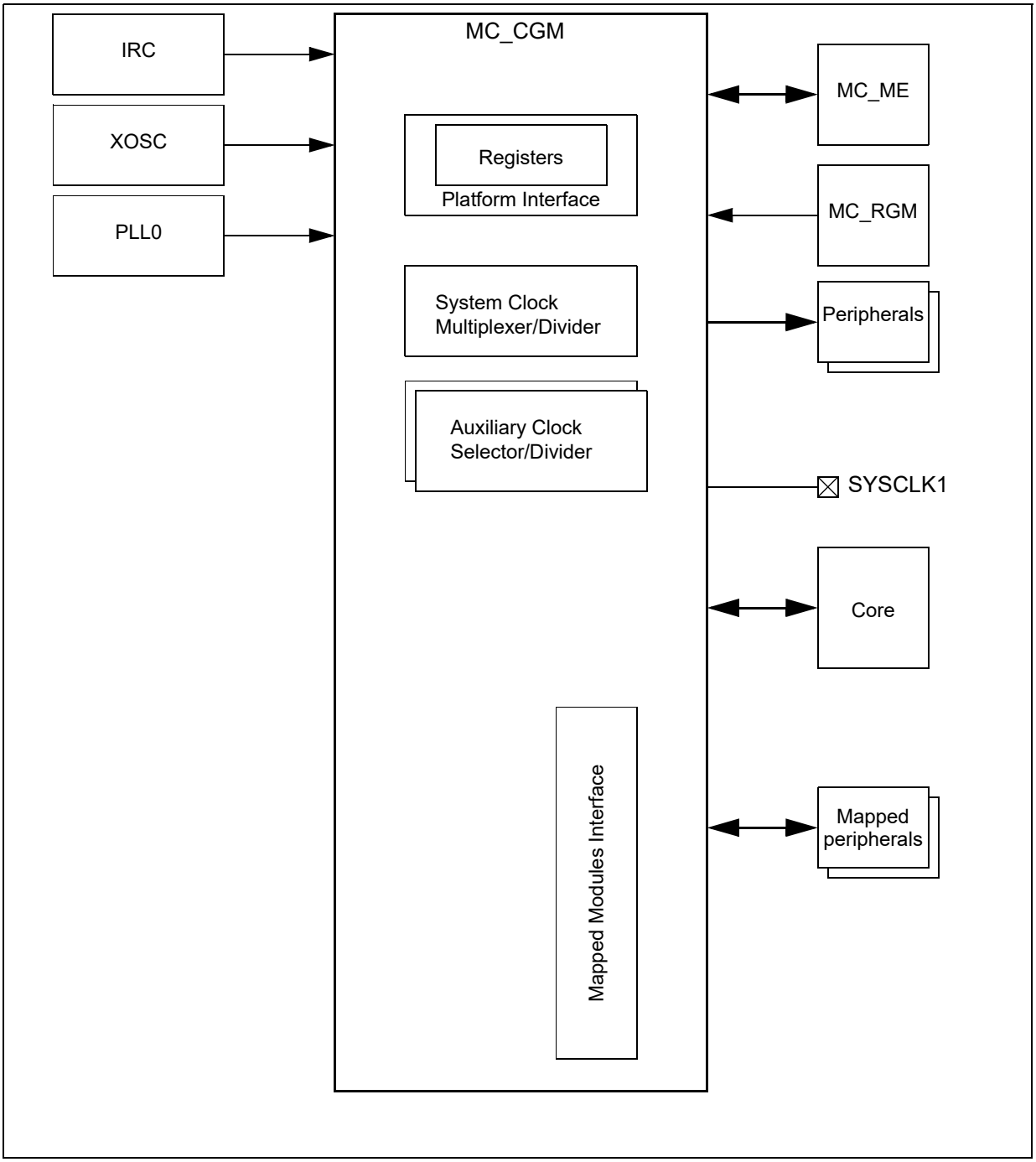
24.1 Introduction

24.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the chip blocks. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see the MC_ME chapter for more details). A set of MC_CGM registers controls the clock dividers which are used for divided system and peripheral clock generation. The memory spaces of system and peripheral clock sources which have addressable memory spaces are accessed through the MC_CGM memory space.

Figure 180 shows the MC_CGM block diagram.

Figure 180. MC_CGMblock diagram



24.1.2 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 the 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

24.2 External signal description

The MC_CGM delivers output clocks to the and SYSCLK1 pins for off-chip use and/or observation.

24.3 Memory map and register definition

Table 231. MC_CGM register description

Offset (Base 0xFFFB_0700)	Name	Size	Access			Location
			User	Supervisor	Test	
0x0000	PCS Switch Duration Register (CGM_PCS_SDUR)	byte	read	read/write	read/write	*on page 449*
0x0004	PCS Divider Change for external crystal oscillator (XOSC) Register (CGM_PCS_DIVC1)	word	read	read/write	read/write	*on page 450*
0x0008	PCS Divider Start for external crystal oscillator (XOSC) Register (CGM_PCS_DIVS1)	word	read	read/write	read/write	*on page 450*
0x000C	PCS Divider End for external crystal oscillator (XOSC) Register (CGM_PCS_DIVE1)	word	read	read/write	read/write	*on page 451*
0x0010	PCS Divider Change for PLL0 PHI Register (CGM_PCS_DIVC2)	word	read	read/write	read/write	*on page 452*
0x0014	PCS Divider Start for PLL0 PHI Register (CGM_PCS_DIVS2)	word	read	read/write	read/write	*on page 452*
0x0018	PCS Divider End for PLL0 PHI Register (CGM_PCS_DIVE2)	word	read	read/write	read/write	*on page 453*
0x001C	PCS Divider Change for PLL0 PHI1 Register (CGM_PCS_DIVC3)	word	read	read/write	read/write	*on page 454*

Table 231. MC_CGM register description(Continued)

Offset (Base 0xFFFB_0700)	Name	Size	Access			Location
			User	Supervisor	Test	
0x0020	PCS Divider Start for PLL0 PHI1 Register (CGM_PCS_DIVS3)	word	read	read/write	read/write	*on page 454*
0x0024	PCS Divider End for PLL0 PHI1 Register (CGM_PCS_DIVE3)	word	read	read/write	read/write	*on page 455*
0x00E4	System Clock Select Status Register (CGM_SC_SS)	word	read	read	read	*on page 456*
0x00E8	System Clock Divider Configuration 0 Register (CGM_SC_DC0)	word	read	read/write	read/write	*on page 457*
0x00EC	System Clock Divider Configuration 1 Register (CGM_SC_DC1)	word	read	read/write	read/write	*on page 457*
0x0100	Aux Clock 0 Select Control Register (CGM_AC0_SC)	word	read	read/write	read/write	*on page 458*
0x0104	Aux Clock 0 Select Status Register (CGM_AC0_SS)	word	read	read	read	*on page 459*
0x0108	Aux Clock 0 Divider Configuration 0 Register (CGM_AC0_DC0)	word	read	read/write	read/write	*on page 460*
0x010C	Aux Clock 0 Divider Configuration 1 Register (CGM_AC0_DC1)	word	read	read/write	read/write	*on page 461*
0x0110	Aux Clock 0 Divider Configuration 2 Register (CGM_AC0_DC2)	word	read	read/write	read/write	*on page 461*
0x0120	Aux Clock 1 Select Control Register (CGM_AC1_SC)	word	read	read/write	read/write	*on page 462*
0x0124	Aux Clock 1 Select Status Register (CGM_AC1_SS)	word	read	read	read	*on page 463*
0x0128	Aux Clock 1 Divider Configuration 0 Register (CGM_AC1_DC0)	word	read	read/write	read/write	*on page 464*
0x0148	Aux Clock 2 Divider Configuration 0 Register (CGM_AC2_DC0)	word	read	read/write	read/write	*on page 465*
0x0160	Aux Clock 3 Select Control Register (CGM_AC3_SC)	word	read	read/write	read/write	*on page 466*
0x0164	Aux Clock 3 Select Status Register (CGM_AC3_SS)	word	read	read	read	*on page 467*
0x01E0	Aux Clock 7 Select Control Register (CGM_AC7_SC)	word	read	read/write	read/write	*on page 468*
0x01E4	Aux Clock 7 Select Status Register (CGM_AC7_SS)	word	read	read	read	*on page 469*
0x01E8	Aux Clock 7 Divider Configuration 0 Register (CGM_AC7_DC0)	word	read	read/write	read/write	*on page 470*
0x0200	Aux Clock 8 Select Control Register (CGM_AC8_SC)	word	read	read/write	read/write	*on page 471*

Table 231. MC_CGM register description(Continued)

Offset (Base 0xFFFB_0700)	Name	Size	Access			Location
			User	Supervisor	Test	
0x0204	Aux Clock 8 Select Status Register (CGM_AC8_SS)	word	read	read	read	on page 472
0x0208	Aux Clock 8 Divider Configuration 0 Register (CGM_AC8_DC0)	word	read	read/write	read/write	on page 473
0x0240	Aux Clock 10 Select Control Register (CGM_AC10_SC)	word	read	read/write	read/write	on page 474
0x0244	Aux Clock 10 Select Status Register (CGM_AC10_SS)	word	read	read	read	on page 475
0x0248	Aux Clock 10 Divider Configuration 0 Register (CGM_AC10_DC0)	word	read	read/write	read/write	on page 476
0x0260	Aux Clock 11 Select Control Register (CGM_AC11_SC)	word	read	read/write	read/write	on page 477
0x0264	Aux Clock 11 Select Status Register (CGM_AC11_SS)	word	read	read	read	on page 478
0x0268	Aux Clock 11 Divider Configuration 0 Register (CGM_AC11_DC0)	word	read	read/write	read/write	on page 479
0x026C	Aux Clock 11 Divider Configuration 1 Register (CGM_AC11_DC1)	word	read	read/write	read/write	on page 480

Note: Any access to unused registers as well as write accesses to read-only registers :

- *does not change register content*
- *causes a transfer error*

24.3.1 Register descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the CGM_PCS_DIVC1 register RATE bits may be accessed as a word at address offset 0x0704, as a half-word at address offset 0x0706, or as a byte at address offset 0x0707.

24.3.1.1 PCS Switch Duration Register (CGM_PCS_SDUR)

This register contains the progressive system clock switching duration for each step. See *Section 24.4.1.2, Progressive System Clock Switching* for details on how to set this value.

Address 0x0700 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R W	SDUR							
Reset	0	0	0	0	0	0	0	0

Figure 181. PCS Switch Duration Register (CGM_PCS_SDUR)

Table 232. PCS Switch Duration Register (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 (IRCOSC) cycles.

24.3.1.2 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 24.4.1.2, Progressive System Clock Switching* for details on how to set these values.

Note: Byte write accesses are not allowed for the INIT field of this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x0704 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INIT															
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 182. PCS Divider Change Register 1 (CGM_PCS_DIVC1)

Table 233. PCS Divider Change Register 1 (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).
16–23	Reserved
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).

24.3.1.3 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 24.4.1.2, Progressive System Clock Switching* for details on how to set these values.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x0708 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 234. PCS Divider Start Register 1 (CGM_PCS_DIVS1) field descriptions

Field	Description
0–11	Reserved
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).

24.3.1.4 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 24.4.1.2, Progressive System Clock Switching* for details on how to set This value.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x070C 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 235. PCS Divider End Register 1 (CGM_PCS_DIVE1) field descriptions

Field	Description
0–11	Reserved
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).

24.3.1.5 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 PLL0 PHI on ramp-up and ramp-down, respectively. See *Section 24.4.1.2, Progressive System Clock Switching* for details on how to set these values.

Note: Byte write accesses are not allowed for the INIT field of this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x0710 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INIT															
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 185. PCS Divider Change Register 2 (CGM_PCS_DIVC2)

Table 236. PCS Divider Change Register 2 (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 PLL0 PHI.
16–23	Reserved
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 PLL0 PHI.

24.3.1.6 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 PLL0 PHI on ramp-up. See *Section 24.4.1.2, Progressive System Clock Switching* for details on how to set this value.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x0714 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 237. PCS Divider Start Register 2 (CGM_PCS_DIVS2) field descriptions

Field	Description
0–11	Reserved
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 PLL0 PHI.

24.3.1.7 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 PLL0 PHI on ramp-down. See *Section 24.4.1.2, Progressive System Clock Switching* for details on how to set This value.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x0718 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 238. PCS Divider End Register 2 (CGM_PCS_DIVE2) field descriptions

Field	Description
0–11	Reserved
12–31 DIVE	**Divider End Value** — This is the clock divider end value for the clock ramp-down phase when switching from the PLL0 PHI.

24.3.1.8 PCS Divider Change Register 3 (CGM_PCS_DIVC3)

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 PLL0 PHI1 on ramp-up and ramp-down, respectively. See *Section 24.4.1.2, Progressive System Clock Switching* for details on how to set these values.

Note: Byte write accesses are not allowed for the INIT field of this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x071C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INIT															
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 188. PCS Divider Change Register 3 (CGM_PCS_DIVC3)

Table 239. PCS Divider Change Register 3 (CGM_PCS_DIVC3) 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 reserved.
16–23	Reserved
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 reserved.

24.3.1.9 PCS Divider Start Register 3 (CGM_PCS_DIVS3)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL0 PHI1 on ramp-up. See *Section 24.4.1.2, Progressive System Clock Switching* for details on how to set this value.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x0720 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 3 (CGM_PCS_DIVS3)

Table 240. PCS Divider Start Register 3 (CGM_PCS_DIVS3) field descriptions

Field	Description
0–11	Reserved
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 reserved.

24.3.1.10 PCS Divider End Register 3 (CGM_PCS_DIVE3)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL0 PHI1 on ramp-down. See *Section 24.4.1.2, Progressive System Clock Switching* for details on how to set This value.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x0724 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 3 (CGM_PCS_DIVE3)

Table 241. PCS Divider End Register 3 (CGM_PCS_DIVE3) field descriptions

Field	Description
0–11	Reserved
12–31 DIVE	**Divider End Value** — This is the clock divider end value for the clock ramp-down phase when switching from the reserved.

24.3.1.11 System Clock Select Status Register (CGM_SC_SS)

This register provides the current system clock source selection.

Address 0x07E4 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 191. System Clock Select Status Register (CGM_SC_SS)

Table 242. System Clock Select Status Register (CGM_SC_SS) field descriptions

Field	Description
0–3	Reserved
4–7 SELSTAT	**System Clock Source Selection Status** — This value indicates the current source for the system clock. 0000 16 MHz internal RC oscillator (IRCOSC) 0001 external crystal oscillator (XOSC) 0010 PLL0 PHI 0011 PLL0 PHI1 0100 reserved 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
8–11	Reserved
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 (IRCOSC) due to SAFE mode request or reset succeeded 101 switch to 16 MHz internal RC oscillator (IRCOSC) due to SAFE mode request or reset succeeded, but current clock source was inactive 110 reserved 111 reserved

Table 242. System Clock Select Status Register (CGM_SC_SS) field descriptions(Continued)

Field	Description
15 SWIP	Switch In Progress 0 clock source switching has completed 1 clock source switching is in progress
16–31	Reserved

24.3.1.12 System Clock Divider 0 Configuration Register (CGM_SC_DC0)

This register controls system clock divider 0. See *Section 24.4.1.4.1, System Clock Divider Synchronization* for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, however the value is not loaded with the new value.

Address 0x07E8 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	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 192. System Clock Divider 0 Configuration Register (CGM_SC_DC0)

Table 243. System Clock Divider 0 Configuration Register (CGM_SC_DC0) 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’.
1–9	Reserved
10–15 DIV	Divider 1 Division Value — The resultant core 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 core clock remains disabled.
16–31	Reserved

24.3.1.13 System Clock Divider 1 Configuration Register (CGM_SC_DC1)

This register controls system clock divider 1. See *Section 24.4.1.4.1, System Clock Divider Synchronization* for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x07EC 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	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 193. System Clock Divider 1 Configuration Register (CGM_SC_DC1)

Table 244. System Clock Divider 1 Configuration Register (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’.
1–9	Reserved
10–15 DIV	Divider 1 Division Value — The resultant AIPS 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 AIPS clock remains disabled.
16–31	Reserved

24.3.1.14 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: peripheral clock
- divided by auxiliary clock 0 divider 1: Sigma-Delta ADC clock
- divided by auxiliary clock 0 divider 2: SAR ADC clock

See *Figure 194* for details.

Address 0x0800 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 194. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

Table 245. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC) field descriptions

Field	Description
0–3	Reserved
4–7 SELCTL	Auxiliary Clock 0 Source Selection Control — Selects the source for auxiliary clock 0. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 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
8–31	Reserved

24.3.1.15 Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

This register provides the current auxiliary clock 0 source selection.

Address 0x0804 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 195. Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

Table 246. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC) field descriptions

Field	Description
0–3	Reserved
4–7 SELSTAT	**Auxiliary Clock 0 Source Selection Status** — This value indicates the current source for auxiliary clock 0. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 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
8–31	Reserved

24.3.1.16 Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

This register controls auxiliary clock 0 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x0808 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 196. Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

Table 247. Auxiliary Clock 0 Divider 0 Configuration Register (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
1–11	Reserved
12–15 DIV	Divider Division Value — The resultant peripheral 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 peripheral clock remains disabled.
16–31	Reserved

24.3.1.17 Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

This register controls auxiliary clock 0 divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x080C 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 197. Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

Table 248. Auxiliary Clock 0 Divider 1 Configuration Register (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
1–8	Reserved
9–15 DIV	Divider Division Value — The resultant Sigma-Delta 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 Sigma-Delta ADC clock remains disabled.
16–31	Reserved

24.3.1.18 Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)

This register controls auxiliary clock 0 divider 2.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x0810 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 198. Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)

Table 249. Auxiliary Clock 0 Divider 2 Configuration Register (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
1–8	Reserved
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 2 is disabled), any write access to the DIV field is ignored and the SAR ADC clock remains disabled.
16–31	Reserved

24.3.1.19 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: LFAST clock

See *Figure 199* for details.

Address 0x0820 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 199. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

Table 250. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC) field descriptions

Field	Description
0–3	Reserved
4–7 SELCTL	**Auxiliary Clock 1 Source Selection Control** — This value selects the current source for auxiliary clock 1. 0000 reserved 0001 XOSC 0010 PLL0 PHI 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31	Reserved

24.3.1.20 Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

This register provides the current auxiliary clock 1 source selection.

Address 0x0824 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 200. Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

Table 251. Auxiliary Clock 1 Select Status Register (CGM_AC1_SS) field descriptions

Field	Description
0–3	Reserved
4–7 SELSTAT	**Auxiliary Clock 1 Source Selection Status** — This value indicates the current source for auxiliary clock 1. 0000 reserved 0001 XOSC 0010 PLL0 PHI 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31	Reserved

24.3.1.21 Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

This register controls auxiliary clock 1 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x0828 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 201. Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

Table 252. Auxiliary Clock 1 Divider 0 Configuration Register (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
1–8	Reserved
9–15 DIV	Divider Division Value — The resultant LFAST 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 LFAST clock remains disabled.
16–31	Reserved

24.3.1.22 Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)

This register controls auxiliary clock 2 divider 0 when the “CAN jitter” feature is not enabled (see *Section 24.4.3.1, Divider Jitter Injection*).

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Address 0x0848 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 202. Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)

Table 253. Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0) field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 2 divider 0 1 Enable auxiliary clock 2 divider 0
1–9	Reserved
10–15 DIV	Divider Division Value — The resultant SENT will have a period 'DIV + 1' times that of auxiliary clock 2. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SENT remains disabled.
16–31	Reserved

24.3.1.23 Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

This register is used to select the current clock source for the PLL0 reference clock.

See *Figure 203* for details.

Address 0x0860 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 203. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

Table 254. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC) field descriptions

Field	Description
0–3	Reserved
4–7 SELCTL	**Auxiliary Clock 3 Source Selection Control** — This value selects the current source for auxiliary clock 3. 0000 IRCOSC 0001 XOSC 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
8–31	Reserved

24.3.1.24 Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

This register provides the current auxiliary clock 3 source selection.

Address 0x0864 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 204. Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

Table 255. Auxiliary Clock 3 Select Status Register (CGM_AC3_SS) field descriptions

Field	Description
0–3	Reserved
4–7 SELSTAT	**Auxiliary Clock 3 Source Selection Status** — This value indicates the current source for auxiliary clock 3. 0000 IRCOSC 0001 XOSC 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
8–31	Reserved

24.3.1.25 Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)

Address 0x08E0 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 205. Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 7 divider 0: SYSCLK1 pin clock

See *Figure 224* for details.

Table 256. Auxiliary Clock 7 Select Control Register (CGM_AC7_SC) field descriptions

Field	Description
0–3	Reserved
4–7 SELCTL	**Auxiliary Clock 7 Source Selection Control** — This value selects the current source for auxiliary clock 7. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31	Reserved

24.3.1.26 Auxiliary Clock 7 Select Status Register (CGM_AC7_SS)

Address 0x08E4 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 206. Auxiliary Clock 7 Select Status Register (CGM_AC7_SS)

This register provides the current auxiliary clock 7 source selection.

Table 257. Auxiliary Clock 7 Select Status Register (CGM_AC7_SS) field descriptions

Field	Description
0–3	Reserved
4–7 SELSTAT	**Auxiliary Clock 7 Source Selection Status** — This value indicates the current source for auxiliary clock 7. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31	Reserved

24.3.1.27 Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)

Address 0x08E8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	DIV								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 207. Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)

This register controls auxiliary clock 7 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Table 258. Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0) field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 7 divider 0 1 Enable auxiliary clock 7 divider 0
1–6	Reserved
7–15 DIV	Divider Division Value — The resultant SYSCLK1 pin clock will have a period 'DIV + 1' times that of auxiliary clock 7. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SYSCLK1 pin clock remains disabled.
16–31	Reserved

24.3.1.28 Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)

Address 0x0900 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 208. Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 8 divider 0: CCCU clock

See *Figure 225* for details.

Table 259. Auxiliary Clock 8 Select Control Register (CGM_AC8_SC) field descriptions

Field	Description
0–3	Reserved
4–7 SELCTL	**Auxiliary Clock 8 Source Selection Control** — This value selects the current source for auxiliary clock 8. 0000 reserved 0001 XOSC 0010 PLL0 PHI 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
8–31	Reserved

24.3.1.29 Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)

Address 0x0904 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 209. Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)

This register provides the current auxiliary clock 8 source selection.

Table 260. Auxiliary Clock 8 Select Status Register (CGM_AC8_SS) field descriptions

Field	Description
0–3	Reserved
4–7 SELSTAT	**Auxiliary Clock 8 Source Selection Status** — This value indicates the current source for auxiliary clock 8. 0000 reserved 0001 XOSC 0010 PLL0 PHI 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
8–31	Reserved

24.3.1.30 Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)

Address 0x0908 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 210. Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)

This register controls auxiliary clock 8 divider 0 when the “CAN jitter” feature is not enabled (see *Section 24.4.3.1, Divider Jitter Injection*).

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Table 261. Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0) field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 8 divider 0 1 Enable auxiliary clock 8 divider 0
1–9	Reserved
10–15 DIV	Divider Division Value — The resultant CCCU clock will have a period 'DIV + 1' times that of auxiliary clock 8. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the CCCU clock remains disabled.
16–31	Reserved

24.3.1.31 Auxiliary Clock 10 Select Control Register (CGM_AC10_SC)

Address 0x0940 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	1	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	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 10 Select Control Register (CGM_AC10_SC)

This register is used to select the current clock source for the for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 10 divider 0: FEC reference clock

See *Figure 226* for details.

Table 262. Auxiliary Clock 10 Select Control Register (CGM_AC10_SC) field descriptions

Field	Description
0–3	Reserved
4–7 SELCTL	**Auxiliary Clock 10 Source Selection Control** — This value selects the current source for auxiliary clock 10. 0000 reserved 0001 XOSC 0010 PLL0 PHI 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31	Reserved

24.3.1.32 Auxiliary Clock 10 Select Status Register (CGM_AC10_SS)

Address 0x0944 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	1	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	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 10 Select Status Register (CGM_AC10_SS)

This register provides the current auxiliary clock 10 source selection.

Table 263. Auxiliary Clock 10 Select Status Register (CGM_AC10_SS) field descriptions

Field	Description
0–3	Reserved
4–7 SELSTAT	**Auxiliary Clock 10 Source Selection Status** — This value indicates the current source for auxiliary clock 10. 0000 reserved 0001 XOSC 0010 PLL0 PHI 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31	Reserved

24.3.1.33 Auxiliary Clock 10 Divider 0 Configuration Register (CGM_AC10_DC0)

Address 0x0948 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 213. Auxiliary Clock 10 Divider 0 Configuration Register (CGM_AC10_DC0)

This register controls auxiliary clock 10 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not be loaded with the new value.

Table 264. Auxiliary Clock 10 Divider 0 Configuration Register (CGM_AC10_DC0) field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 10 divider 0 1 Enable auxiliary clock 10 divider 0
1–11	Reserved
12–15 DIV	Divider Division Value — The resultant FEC reference clock will have a period 'DIV + 1' times that of auxiliary clock 10. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the FEC reference clock remains disabled.
16–31	Reserved

24.3.1.34 Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)

Address 0x0960 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 214. Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)

This register is used to select the current clock source for the for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 11 divider 0: DSPI clock 0
- divided by auxiliary clock 11 divider 1: DSPI clock 1

See *Figure 225* for details.

Table 265. Auxiliary Clock 11 Select Control Register (CGM_AC11_SC) field descriptions

Field	Description
0–3	Reserved
4–7 SELCTL	**Auxiliary Clock 11 Source Selection Control** — This value selects the current source for auxiliary clock 11. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31	Reserved

24.3.1.35 Auxiliary Clock 11 Select Status Register (CGM_AC11_SS)

Address 0x0964 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	1	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	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 215. Auxiliary Clock 11 Select Status Register (CGM_AC11_SS)

This register provides the current auxiliary clock 11 source selection.

Table 266. Auxiliary Clock 11 Select Status Register (CGM_AC11_SS) field descriptions

Field	Description
0–3	Reserved
4–7 SELSTAT	**Auxiliary Clock 11 Source Selection Status** — This value indicates the current source for auxiliary clock 11. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31	Reserved

24.3.1.36 Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)

Address 0x0968 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	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	DIV_FMT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 216. Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)

This register controls auxiliary clock 11 divider 0. See *Section 24.4.3.2: Fractional Divider Details* for divider characteristics.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Table 267. Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0) field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 11 divider 0 1 Enable auxiliary clock 11 divider 0
1–2	Reserved
3–15 DIV	Divider Division Value — The resultant DSPI clock 0 will have a period 'DIV + 1' times that of auxiliary clock 11. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the DSPI clock 0 remains disabled.
16–29	Reserved
30–31 DIV_FMT	Divider Value Format — This value indicates the format of the division value for calculating the division factor 00 Division factor = (DIV + 1) / 1 01 Division factor = (DIV + 1) / 10 10 Division factor = (DIV + 1) / 100 11 Division factor = (DIV + 1) / 1,000

24.3.1.37 Auxiliary Clock 11 Divider 1 Configuration Register (CGM_AC11_DC1)

Address 0x096C 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 217. Auxiliary Clock 11 Divider 1 Configuration Register (CGM_AC11_DC1)

This register controls auxiliary clock 11 divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such accesses do not result in an exception, but the value is not loaded with the new value.

Table 268. Auxiliary Clock 11 Divider 1 Configuration Register (CGM_AC11_DC1) field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 11 divider 1 1 Enable auxiliary clock 11 divider 1
1–11	Reserved

Table 268. Auxiliary Clock 11 Divider 1 Configuration Register (CGM_AC11_DC1) field descriptions(Continued)

Field	Description
12–15 DIV	Divider Division Value — The resultant DSPI clock 1 will have a period 'DIV + 1' times that of auxiliary clock 11. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the DSPI clock 1 remains disabled.
16–31	Reserved

24.4 Functional Description

24.4.1 System Clock Generation

The MC_ME provides the system clock select and switch mask (see MC_ME chapter for more details), and the MC_RGM provides the safe clock request (see MC_RGM chapter for more details). The safe clock request forces the selector to select the 16 MHz internal RC oscillator (IRCOSC) as the system clock and to ignore the system clock select.

24.4.1.1 System Clock Source Selection

During normal operation, the system clock selection is controlled

- on a SAFE mode or reset event, by the MC_RGM
- otherwise, by the MC_ME

24.4.1.2 Progressive System Clock Switching

In order to prevent sudden voltage drops and overshoots due to frequency and load changes, the MC_ME requests the MC_CGM to ramp the system clock frequency 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.

24.4.1.2.1 Generic Clock Change Requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the maximum allowed frequency change rate a_{max} is given in *Equation 8*.

Equation 8

$$a_{max} = \frac{f_{chg}}{f_{src}}$$

24.4.1.2.2 Configuration of CGM_PCS_SDUR

The switch duration field CGM_PCS_SDUR[SDUR] defines the duration of one system clock switch step in terms of 16 MHz internal RC oscillator (IRCOSC). After the expiration of

this time, the module changes the clock divider value which changes the frequency of the system clock.

24.4.1.2.3 Configuration of CGM_PCS_DIVC*n*[RATE]

For a given maximum allowed frequency change rate a_{max} the value d to be programmed into the PCS_DIVCn[RATE] register is given in *Table 269*.

Table 269. 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

24.4.1.2.4 Generic Clock Change Properties

The number *k* of required steps to reach or leave the divided system clock with frequency f_{div} from the source clock with frequency f_{src} is given in *Equation 9*.

Equation 9

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(1 - \frac{f_{src}}{16MHz}\right)}{d}} \right\rceil$$

24.4.1.2.5 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].

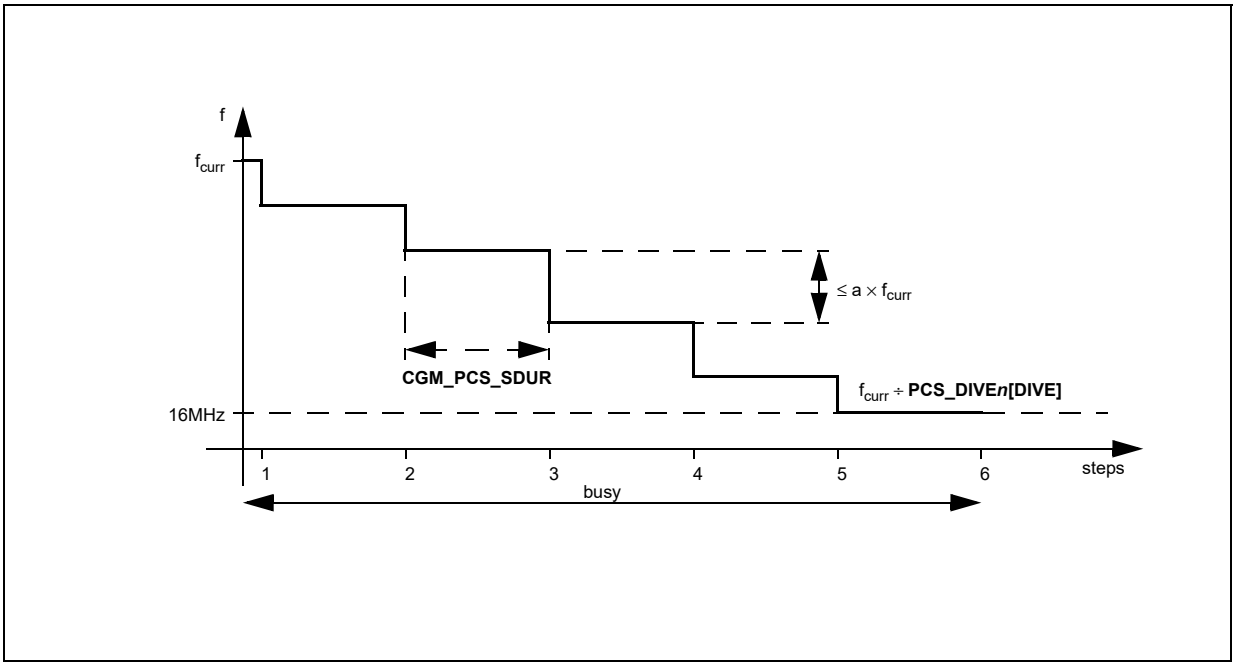
The divider end value for clock ramp-down is given in *Equation 10*.

Equation 10

$$\textbf{PCS_DIVEn[DIVE]} = \frac{f_{src}}{16MHz} \times 1000 - 1$$

Where f_{curr} is the frequency of the currently selected system clock source.

Figure 218. MC_CGM System Clock Ramp-Down Timing (k = 6 example)



24.4.1.2.6 Clock Ramp-Up

The clock ramp-up starts with the given divider value PCS_DIVSn[DIVS] and with the given divider decrement value PCD_DIVCn[INIT] and ends with the divider value 1.

The initial divider change start value PCS_DIVCn[INIT] for system clock ramp-up is given in *Equation 11*.

Equation 11

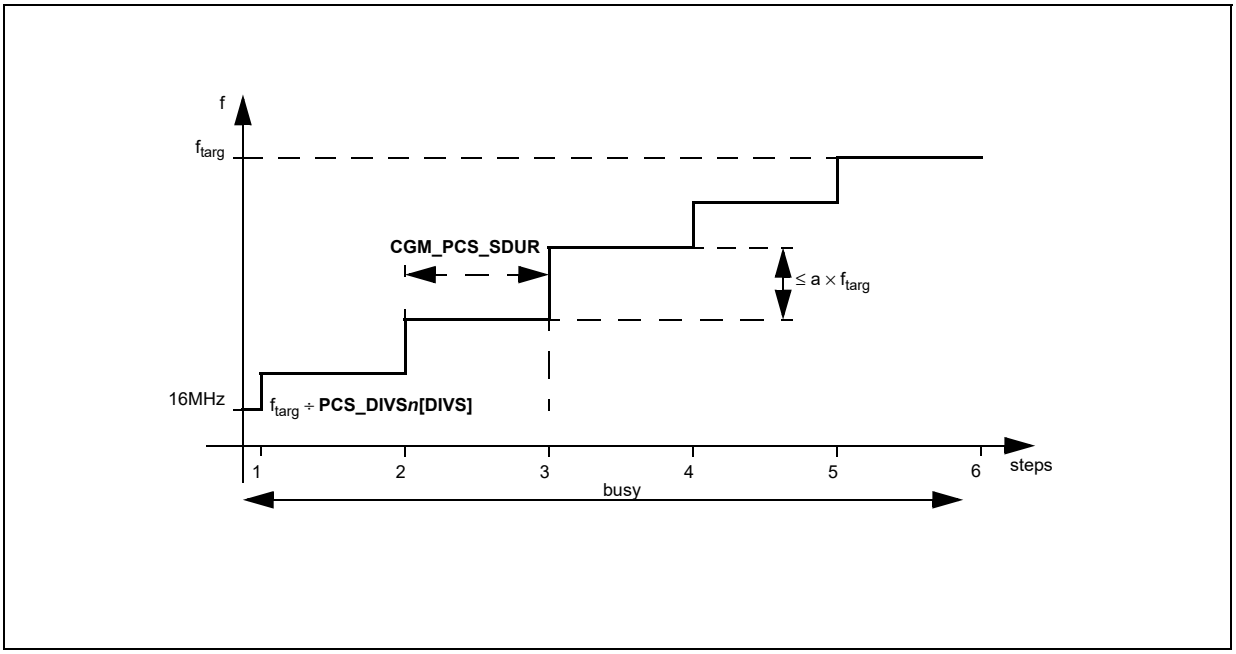
$$\textbf{PCS_DIVCn[INIT]} = d \times k \times 1000 - 1$$

Where k is calculated from *Equation 9* using the target system clock source frequency f_{targ}. The divider start value for clock ramp-up is given in *Equation 12*.

Equation 12

$$\textbf{PCS_DIVSn[DIVS]} = \left(1 + d\frac{k(k+1)}{2}\right) \times 1000 - 1$$

Figure 219. MC_CGM System Clock Ramp-Up Timing (k = 6 example)



24.4.1.3 System Clock Disable

During the TEST mode, the system clock can be disabled by the MC_ME.

24.4.1.4 System Clock Dividers

The MC_CGM generates the following derived clocks from the system clock:

- core clock - controlled by the CGM_SC_DC0 register
- AIPS clock - controlled by the CGM_SC_DC1 register
- AIPS clock - controlled by the CGM_SC_DC2 register

24.4.1.4.1 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.

Example 1. Correct System Clock Divider Configuration

Table 270. MC_CGM Example System Clock Dividision 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. Also, system clock divider 3 has a division factor that is greater than those of both system clock dividers 0 and 1. Since 6 is an integer multiple of 1 and 6 is an integer multiple of 2, its configuration is also correct.

Example 2. Incorrect System Clock Divider Configuration

Table 271. MC_CGM Example System Clock Dividision Values Incompatible with Divider Synchronization

Divider	Division Factor	Register Value
0	1	**CGM_SC_DC0[DIV]** = 0
1	4	**CGM_SC_DC1[DIV]** = 3
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 4 is an integer multiple of 1, its configuration is correct. Also, system clock divider 3 has a division factor that is greater than those of both system clock dividers 0 and 1. Since 6 is not an integer multiple of 4, its configuration is incorrect.

Warning: 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.

24.4.1.4.2 System Clock Divider Update Triggering

As opposed to the auxiliary clock dividers which update immediately on a configuration change, the system clock dividers are updated only with the next software triggered mode change. This is to ensure 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 the CGM_SC_DC0…2 registers as desired and aligned with the target system clock source frequency making sure that the dividers are configured consistently as described in *Section 24.4.1.4.1, System Clock Divider Synchronization*.
3. Request a mode change in the MC_ME to the desired target mode.
4. When mode change has completed and the system clock switching has been successful, the new system clock divider configuration will be in effect.

See the MC_ME chapter for details on how to configure modes and make mode change requests.

24.4.2 Auxiliary Clock Generation

Figure 220 through *Figure 226* show the block diagrams of the generation logic for the various auxiliary clocks. See the following sections for auxiliary clock selection control:

- *Section 24.3.1.14: Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)*
- *Section 24.3.1.19: Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)*
- *Section 24.3.1.23: Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)*
- *Section 24.3.1.25: Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)*
- *Section 24.3.1.28: Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)*
- *Section 24.3.1.31: Auxiliary Clock 10 Select Control Register (CGM_AC10_SC)*
- *Section 24.3.1.34: Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)*

Figure 220. MC_CGM Auxiliary Clock 0 Generation Overview

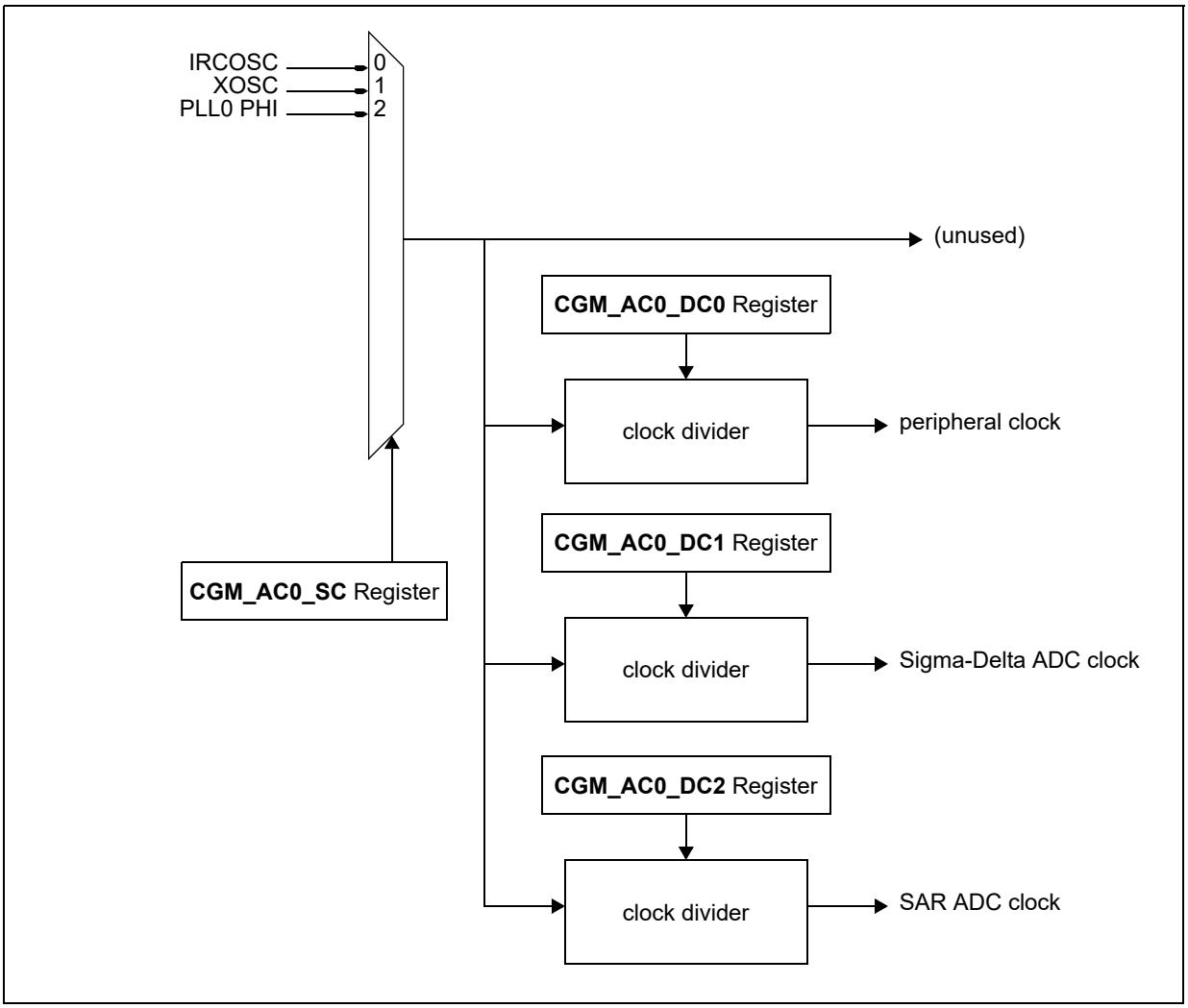


Figure 221. MC_CGM Auxiliary Clock 1 Generation Overview

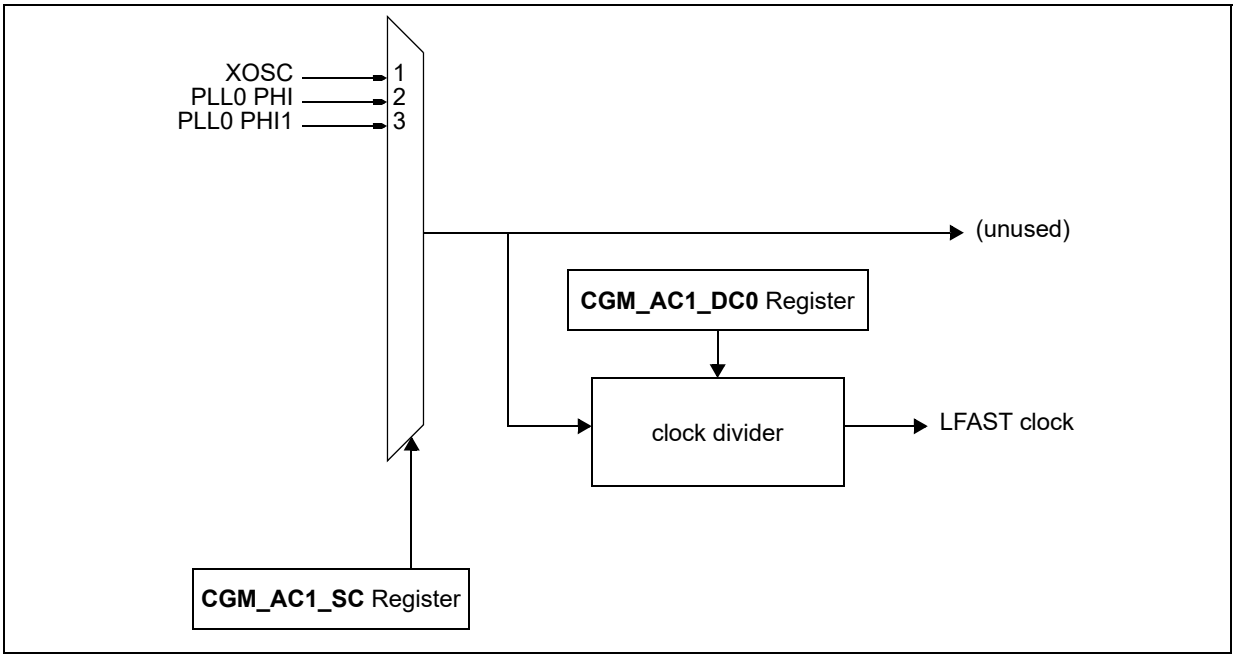


Figure 222. MC_CGM Auxiliary Clock 2 Generation Overview

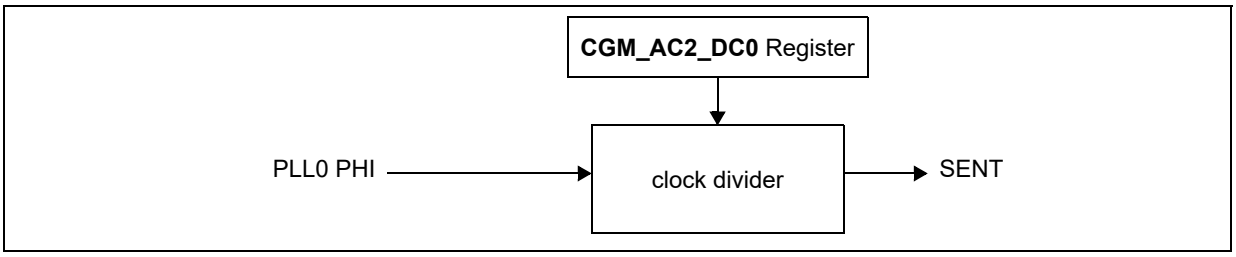


Figure 223. MC_CGM Auxiliary Clock 3 Generation Overview

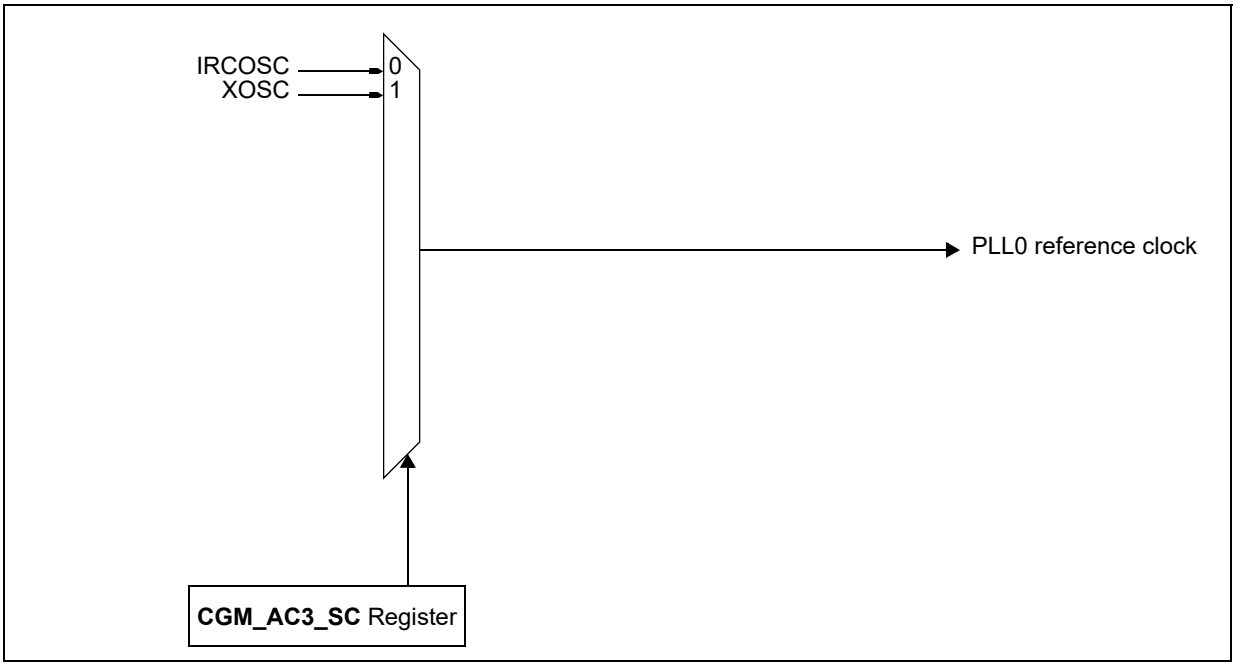


Figure 224. MC_CGM Auxiliary Clock 7 Generation Overview

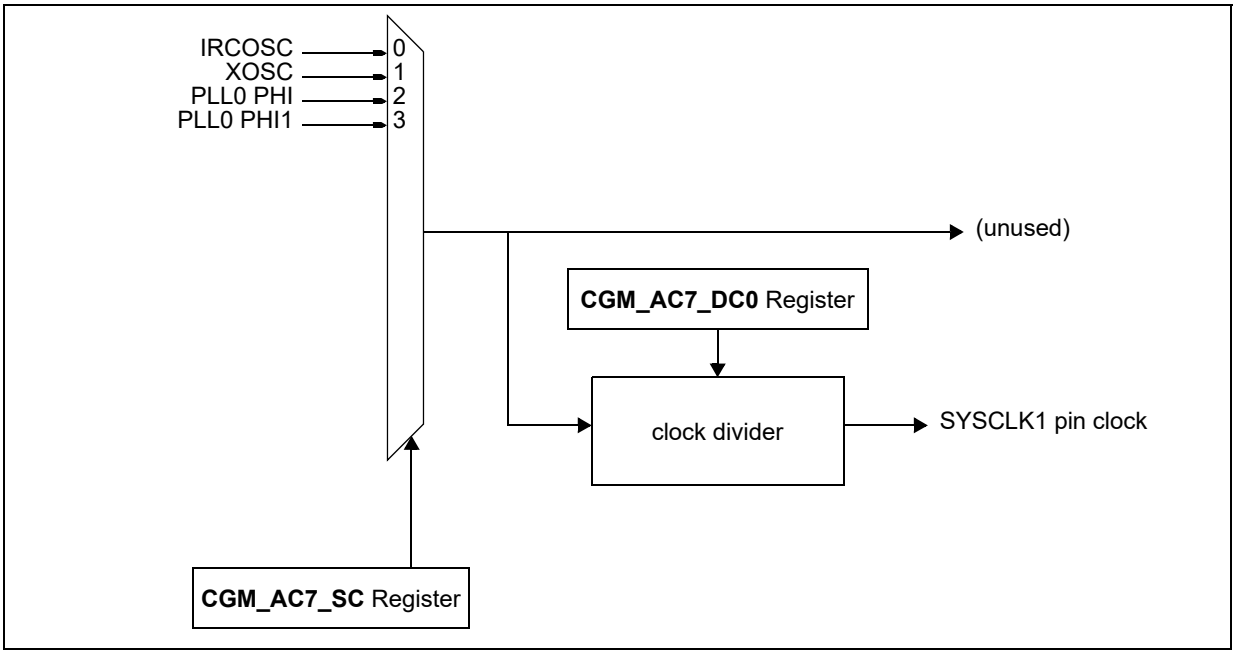


Figure 225. MC_CGM Auxiliary Clock 8 Generation Overview

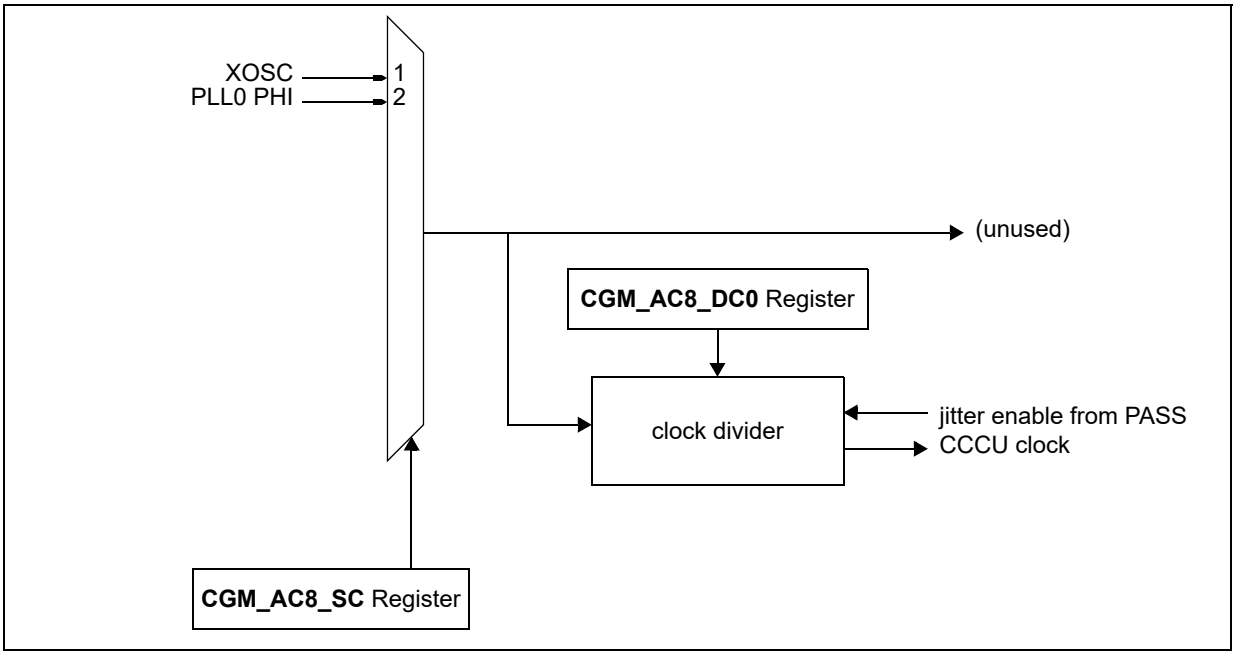


Figure 226. MC_CGM Auxiliary Clock 10 Generation Overview

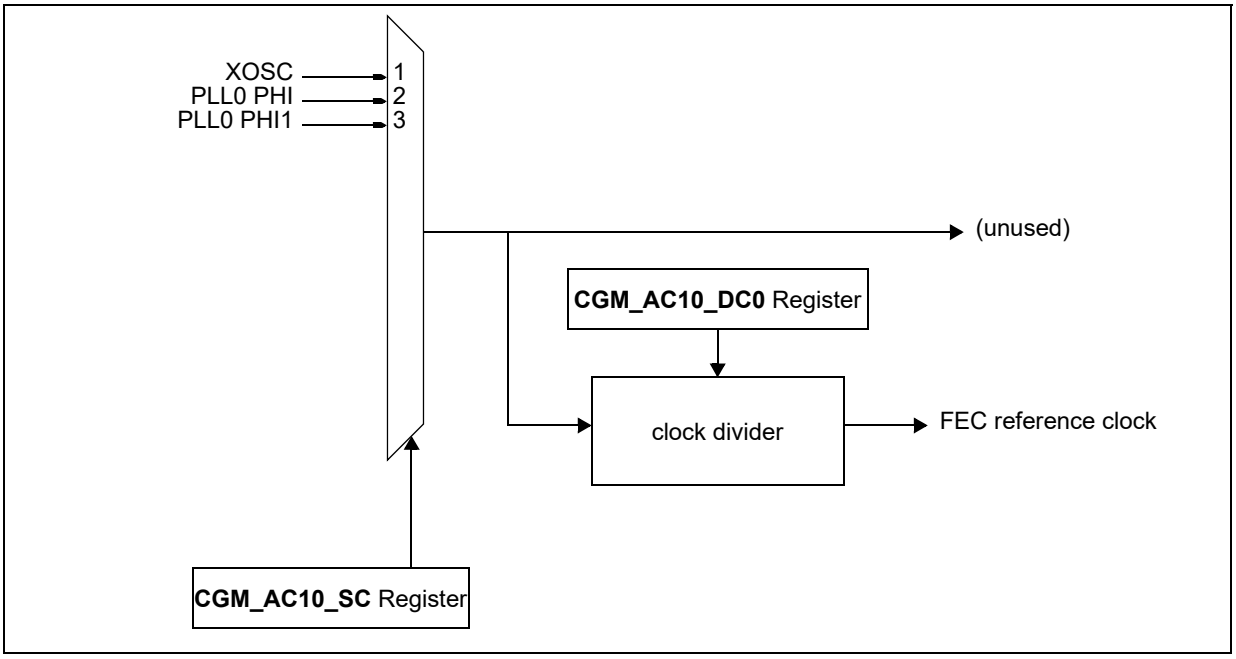
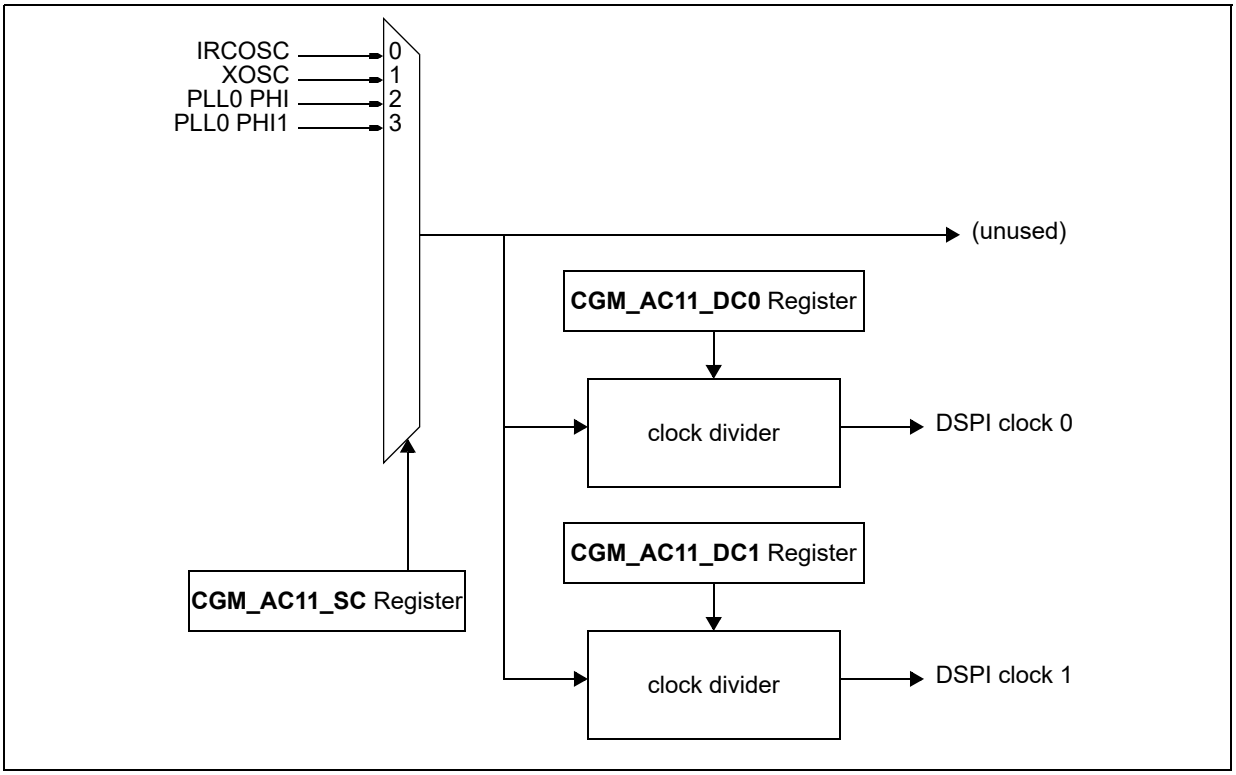


Figure 227. MC_CGM Auxiliary Clock 11 Generation Overview



24.4.2.1 Auxiliary Clock Dividers

The MC_CGM generates the following derived clocks:

- peripheral clock - controlled by the CGM_AC0_DC0 register
- Sigma-Delta ADC clock - controlled by the CGM_AC0_DC1 register
- SAR ADC clock - controlled by the CGM_AC0_DC2 register
- LFAST clock - controlled by the CGM_AC1_DC0 register
- SENT - controlled by the CGM_AC2_DC0 register
- SYSCLK1 pin clock - controlled by the CGM_AC7_DC0 register
- CCCU clock - controlled by the CGM_AC8_DC0 register
- FEC reference clock - controlled by the CGM_AC10_DC0 register
- DSPI clock 0 - controlled by the CGM_AC11_DC0 register
- DSPI clock 1 - controlled by the CGM_AC11_DC1 register

24.4.3 Dividers Functional Description

Dividers are used for the generation of divided system and peripheral clocks. The MC_CGM has the following control registers for built-in dividers:

- *Section 24.3.1.12, System Clock Divider 0 Configuration Register (CGM_SC_DC0)*
- *Section 24.3.1.13, System Clock Divider 1 Configuration Register (CGM_SC_DC1)*
- *Section 24.3.1.16, Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)*
- *Section 24.3.1.17, Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)*
- *Section 24.3.1.18, Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)*
- *Section 24.3.1.21, Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)*
- *Section 24.3.1.22, Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)*
- *Section 24.3.1.27, Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)*
- *Section 24.3.1.30, Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)*
- *Section 24.3.1.33, Auxiliary Clock 10 Divider 0 Configuration Register (CGM_AC10_DC0)*
- *Section 24.3.1.36, Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)*
- *Section 24.3.1.37, Auxiliary Clock 11 Divider 1 Configuration Register (CGM_AC11_DC1)*

The reset value of all counters is ‘1’. If a divider has its DE bit in the respective configuration register set to ‘0’ (the divider is disabled), any value in its DIV field is ignored.

24.4.3.1 Divider Jitter Injection

The clock divider for the CCCU clock injects a jitter so that off-chip communication via the associated peripherals is not possible. For details on enabling this feature, see the “Production Disable” and “CAN Jitter Enable” sections of the PASS chapter.

24.4.3.2 Fractional Divider Details

The clock divider used to generate the DSPI clock 0 divides the selected clock source frequency by a fractional value given by the DIV and DIV_FMT fields of the CGM_AC5_DC0 register.

Equation 13

$$f_{\text{divided clock}} = f_{\text{source clock}} \div ((\mathbf{DIV} + 1) \div 10^{\mathbf{DIV_FMT}})$$

The divided clock frequency is actually an average frequency, since the divider uses only the available rising and falling edges of the source clock. Therefore, the actual frequency switches between two frequencies which are the two frequencies closest to the desired

frequency (above and below) which can be extracted by a division factor *n*/2, where *n* is an integer greater than 1. The switching between the two frequencies is such that the resultant clock's rising edge never deviates from the ideal clock's rising edge by more than 25% of the source clock period and that the resultant average frequency over **DIV** + 1 source clock periods (= 10^{DIV_FMT} divided clock periods) is the desired frequency.

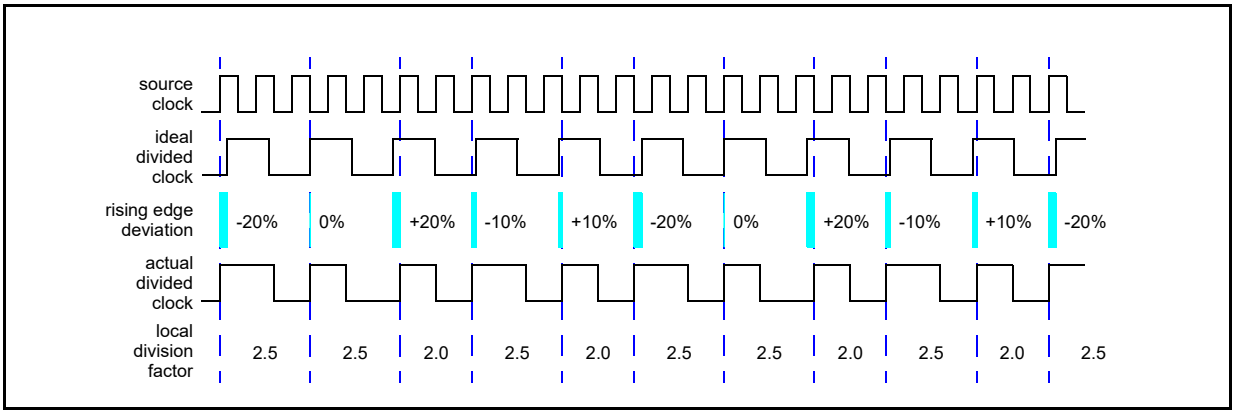
Example 3. Fractional Clock Division

Desired division factor = 2.3

DIV = 22

DIV_FMT = 1

Figure 228. MC_CGM Fractional Division Example Waveform (divide by 2.3)



Equation 14

$$f_{\text{divided clock}}\big|_{max} = f_{\text{source clock}} \div 2.0$$

Equation 15

$$f_{\text{divided clock}}\big|_{min} = f_{\text{source clock}} \div 2.5$$

Equation 16

$$f_{\text{divided clock}}\big|_{ave} = f_{\text{source clock}} \div \frac{((2.5 \times 6) + (2.0 \times 4))}{10}$$

Therefore, the rising edge of the actual divided clock is 'snapped' to the rising or falling edge of the source clock closest to the rising edge of the ideal divided clock (see *Figure 228*). This results in a theoretical maximum deviation, or jitter, of ±25% of the source clock period. In this particular case, the maximum deviation is ±20% of the source clock period.

Equation 17

$$f_{\text{divided clock}}\big|_{\text{ave}} = f_{\text{source clock}} \div 2.3$$

25 OSC Digital Interface (XOSC)

25.1 Introduction

The XOSC digital interface is used to control the on-chip oscillator (XOSC) and provides the register interface with programmable features. Selection of the XOSC as a source clock is controlled by 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

25.2 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 is used as an input to PLL and can be selected as the reference clock for many of the peripherals on the device.

Table 272 shows the possible configurations of the XOSC.

Table 272. XOSC configurations

OSC EN	XOSC_CTL[OSCBYP]	XTAL	EXTAL	XOSC Output	XOSC MODE
0	0	Crystal, or left opened	Crystal, or left opened	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

25.2.1 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].

25.2.2 Oscillator startup delay

A bit in the UTEST flash memory determines whether the analog portion of the oscillator is enabled at powerup. If enabled, the analog portion of the oscillator is powered-up during Phase3 of the device reset sequence (pre-self-test 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).

The default value for this field after reset depends on implementation and can be found in the XOSC Device Configuration. 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.

25.2.3 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).

25.2.4 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.

25.2.4.1 Full Swing Pierce mode

Full Swing Pierce mode (FSP) has the following attributes when compared with LCP:

- Provides a wider frequency range
- Less susceptible to noise
- The crystal amplitude is rail-to-rail
  - Distorted oscillation with several harmonics
  - Increased EMI
- Slower settling time
- Feedback resistor and 300 ohms of current limitation are already integrated.

Note: Consult with crystal supplier to determine the need of extra resistance in XTAL to avoid crystal overdrive.

25.3 Memory map and register definition

Table 273 shows the memory map of the XOSC registers.

Table 273. XOSC memory map

Offset	Register	Location
00h	XOSC_CTL—XOSC Control Register	*Section 25.3.1.1*

25.3.1 Register descriptions

25.3.1.1 XOSC control register (XOSC_CTL)

Offset 00h 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	—(2)	—(2)	—(2)	—(2)	—(2)	I_OSC	0	0	0	0	0	0	0
W																
Reset	–	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See the Clocking chapter for reset value.
2. These bits can be written with any value, but writes are ignored. A read returns last written value.

Figure 229. XOSC_CTL register

Table 274. 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.
1:7	Reserved
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.
17:23	Reserved

Table 274. XOSC_CTL register field descriptions(Continued)

Field	Description
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.
25:31	Reserved

26 IRCOSC digital interface

26.1 Introduction

The Internal RC Oscillator digital interface (IRCOSC) controls the internal 16 MHz RC oscillator system. This oscillator system includes the main internal RC oscillator (MRC), as well as an internal temperature sensor and an internal voltage regulator used to compensate external variations in temperature and voltage. It also provides access to trimming information used for safety implementations.

26.2 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 is initialized from TEST flash memory:

- Native trimming bits
- Fine trimming bits
- TSENS code bits

For safety reasons, these bits can be accessed through the IRCOSC Native Trimming register (NT). The main IRCOSC output frequency can be trimmed by configuring IRCOSC_CTL[USER_TRIM[4:0]]. The USER_TRIM bits can be programmed to modify internal capacitor/resistor values to match output clock frequency with dfvar_SW, as defined in the Data Sheet.

26.3 Memory map and register definition

Table 275 shows the memory map of the IRCOSC registers.

Table 275. IRCOSC memory map

Offset (hex)	Register	Access	Reset Value (hex)	Location
00	IRCOSC Control Register (IRCOSC_CTL)	R/W	0000_0000	*26.3.1.1/499*
04	IRCOSC Native Trimming Register (IRCOSC_NT)	R	0000_0yyy(1)	*26.3.1.2/500*
08	IRCOSC Temperature Trimming Register (IRCOSC_TT)	R	yyyy_yyyy(1)	*26.3.1.3/501*

1. See the “Clocking” chapter for specific reset values.

26.3.1 Register descriptions

26.3.1.1 IRCOSC Control register (IRCOSC_CTL)

The IRCOSC_CTL contains user programmable parameters.

Offset 00h 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	—(3)	0	0	0	USER_TRIM(1)				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	—(3)	—(3)	—(3)	—(3)	—(3)	—(3)	0	—(2)	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.

2 This field may sometimes have the value of 1, and can be cleared by writing a 1 to this bit. However, this field has no effect on the IRCOSC or chip behavior.

3 These bits can be written with any value, but writes are ignored. A read returns last written value.

Figure 230. IRCOSC Control register (IRCOSC_CTL)

Table 276. IRCOSC_CTL field descriptions

Field	Description
0:10	Reserved.
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 277* shows how the IRCOSC_CTL[USER_TRIM] field is used to trim the IRCOSC frequency.
16:31	Reserved

Table 277. IRCOSC_CTL[USER_TRIM] frequency trimming calculation

USER_TRIM[4:0] value	Frequency
....	....
10011	$f + (3 \times \delta f_{TRIM})$
10010	$f + (2 \times \delta f_{TRIM})$
10001	$f + \delta f_{TRIM}$
10000	f
00000	f
00001	$f - \delta f_{TRIM}$

Table 277. IRCOSC_CTL[USER_TRIM] frequency trimming calculation(Continued)

USER_TRIM[4:0] value	Frequency
00010	$f - (2 \times \delta f_{TRIM})$
00011	$f - (3 \times \delta f_{TRIM})$
....	....

Note: See the Data Sheet for δf_{TRIM} value and min/max frequency variations.

26.3.1.2 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 04h Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FT_CODE						0	0	TX_CODE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	REG_EN	TSENS_EN	0	0	RCTRIM							
W																
Reset	0	0	0	0	—(1)	—(1)	0	0	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)

1 See Clocking chapter for specific reset values

Figure 231. IRCOSC Native Trimming register (IRCOSC_NT)

Table 278. IRCOSC_NT field descriptions

Field	Description
0:1	Reserved
2:7 FT_CODE	Output from INTERFACE inside RC to check the fine trimming code applied to RC_core
8:9	Reserved
10:15 TX_CODE	Output from TSENS inside RC to check the TSENS code
16:19	Reserved
20 REG_EN	Enables voltage regulator—trimming value for safety check 0 Voltage regulator is not enabled 1 Voltage regulator is enabled

Table 278. IRCOSC_NT field descriptions(Continued)

Field	Description
21 TSENS_EN	Enables temperature sensor—trimming value for safety check 0 Temperature sensor is not enabled 1 Temperature sensor is enabled
22:23	Reserved
24:31 RCTRIM	IRCOSC trimming value for safety check

26.3.1.3 IRCOSC Temperature Trimming register (IRCOSC_TT)

The TT registers contains system trimming values for the IRCOSC that are loaded from TEST flash memory during reset. TT is a read only register.

Offset 08h Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	COLD_TRIM										0	0	HOT_TRIM	
W																
Reset	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	0	0	_(1)	_(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HOT_TRIM										TSENS_TRIM					
W																
Reset	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

1 See Clocking chapter for specific reset values

Figure 232. IRCOSC Temperature Trimming register (IRCOSC_TT)

Table 279. IRCOSC_TT field descriptions

Field	Description
0:1	Reserved
2:11 COLD_TRIM	Cold temperature trimming value for safety check
12:13	Reserved
14:23 HOT_TRIM	Hot temperature trimming value for safety check
24:25	Reserved
26:31 TSENS_TRIM	Ambient temperature reference code value for safety check

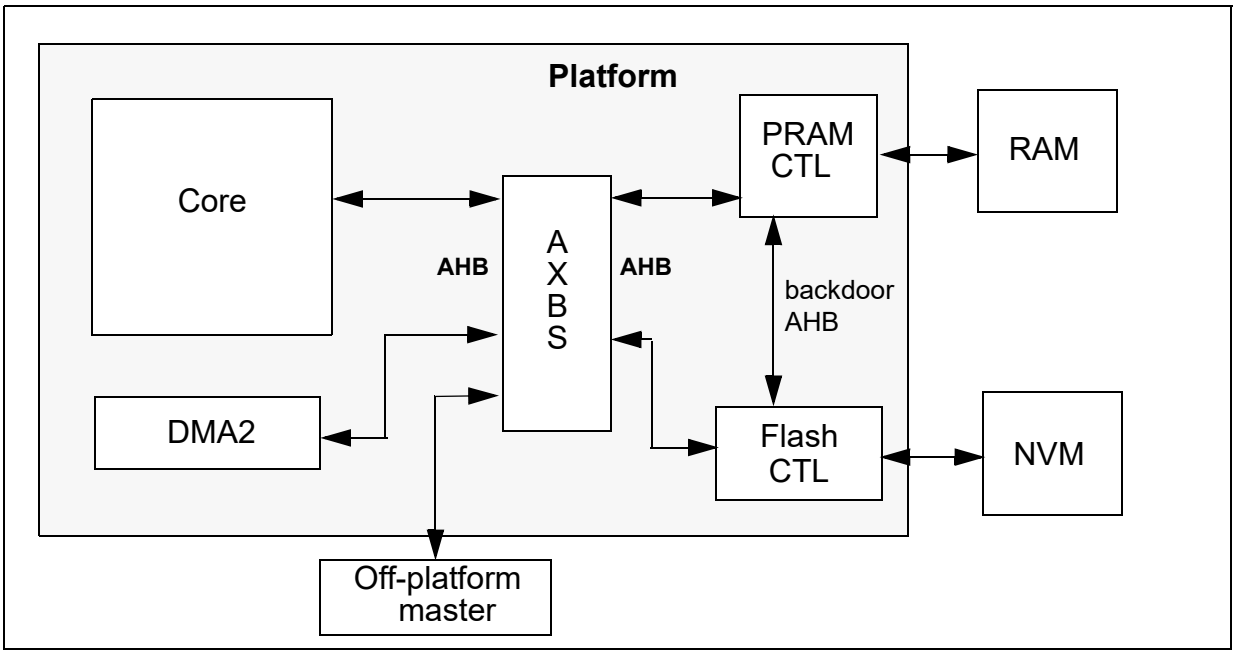
27 RAM controller (PRAM)

27.1 Introduction

This section provides an overview of the Platform RAM Controller. The RAM 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 RAM controller supports two 32-bit AHB interfaces and a 32-bit RAM array interface. The primary AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters. The backdoor AHB port provides a connection from the flash controller to facilitate calibration overlay access. *Figure 233* is a simplified block diagram depicting the position of the RAM controller within a typical platform architecture:

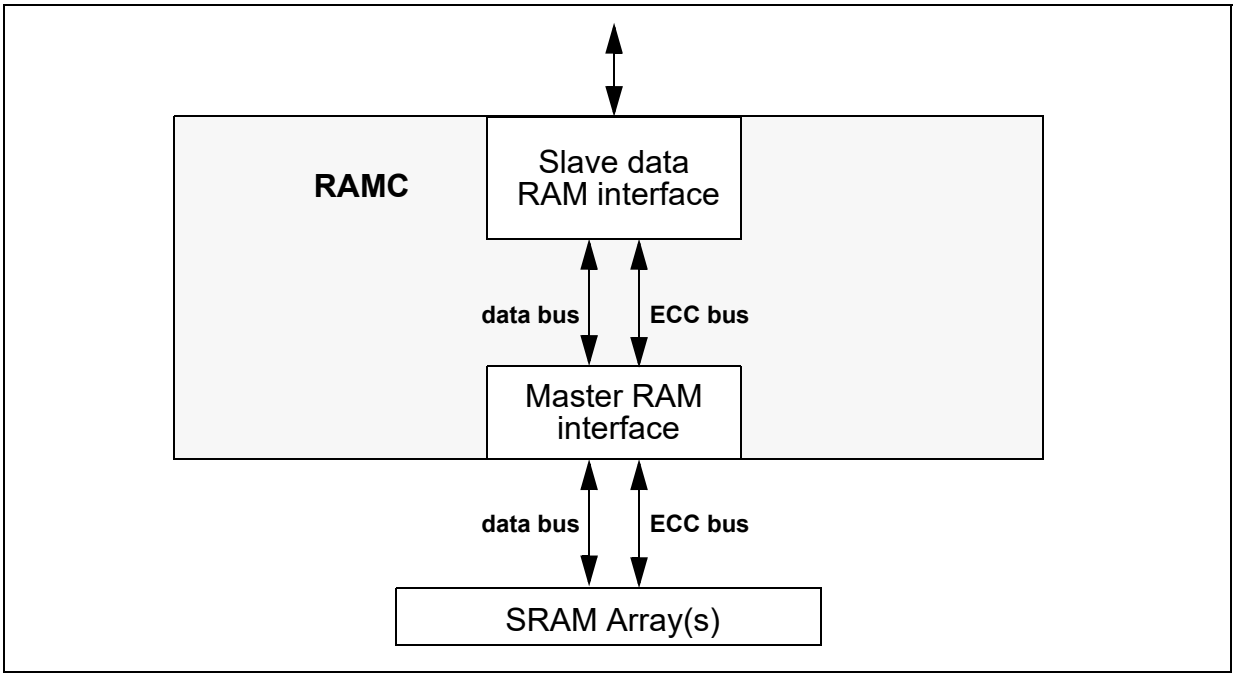
Figure 233. Simplified platform block diagram



The following list summarizes the key features of the RAM controller:

- Supports byte-, halfword- and word-sized reads and writes
- Array interface supports a 32-bit data + 7-bit ECC with SECDED
- Late-write support via 32-bit data + 7-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
- Supports zero wait-state response for all word-sized writes
- Supports a “backdoor” AHB port from platform flash controller

Figure 234. RAM controller block diagram



27.2 Memory map and register definition

The RAM 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 termination.

Table 280. RAM controller memory map

Offset	Register	Size	Access	Mode
0x0000	Platform RAM Configuration Register 1 (PRAMC_PRCR1)	32 bits	R/W	Supr(1)

1. Accessible only in supervisor mode. Accesses in user mode return a transfer error.

27.2.1 Platform RAM Configuration Register 1 (PRAMC_PRCR1)

The Platform RAM Configuration Register 1 (PRCR1) is used to specify operation of the RAM controller.

Note: This register is accessible only in supervisor mode. Accesses in user mode returns a transfer error.

Offset 0x00 (PRCR1) 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	0	0	0	0	0	0	PRI		P1_BO_DIS	P0_BO_DIS	0	0	0	0	0	FT_DIS
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 235. Platform RAM Configuration Register 1 (PRCR1)

Table 281. Platform RAM Configuration Register 1 field descriptions

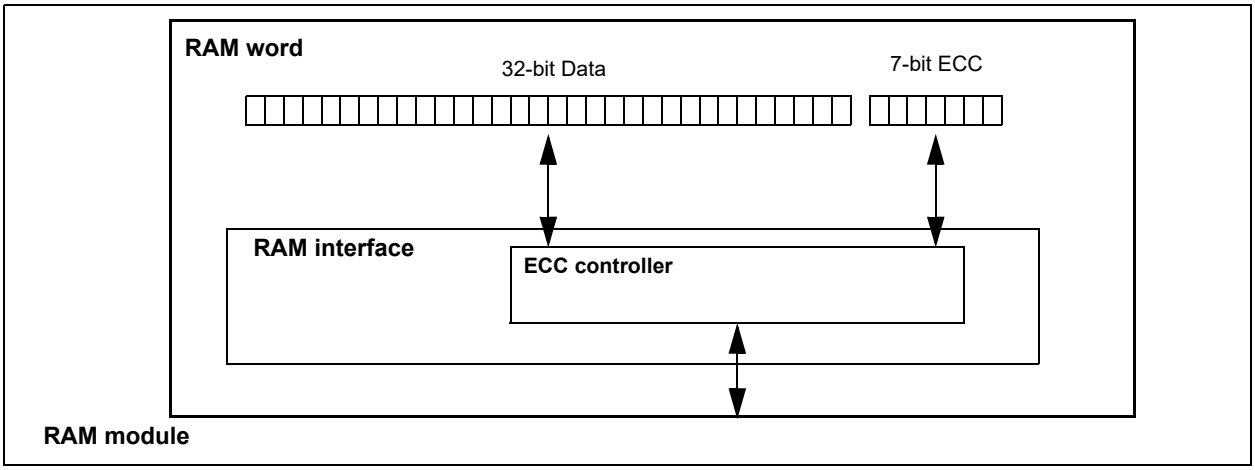
Field	Description
0–21	Reserved
22–23 PRI	AHB port arbitration mode 00 Round robin 01 Port p0 has priority over p1 10 Port p1 has priority over p0 11 Reserved
24 P1_BO_DIS	Port p1 read burst optimization disable 0 32-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response(1) when PRCR[FT_DIS]=1 1 32-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when(1) PRCR[FT_DIS]=1
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
26–30	Reserved
31 FT_DIS	Flow through disabled This field defines the AHB response of the RAM controller on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. 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 PRAM 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 PRAM controller clock frequency.

27.3 SRAM ECC mechanism

The system SRAM width organization at the RAM controller is 32-bit data + 7-bit ECC covering address and data.

Figure 236. RAM word composition



The SRAM ECC detects the following conditions and produces the following results:

- Detects and corrects all 1-bit errors
- Detects and flags all 2-bit errors as non-correctable errors
- Detects 39-bit reads (32-bit data bus plus the 7-bit ECC) that return all zeros or all ones, asserts an error indicator on the bus cycle, and sets the error flag

SRAM does not detect all errors greater than 2 bits.

27.4 Functional description

27.4.1 Read / Write introduction

The RAM controller generates chip-select and write-enable, the array address, and write data as inputs to the RAM array. The RAM 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 RAM controller can optionally incur a single wait state on the AHB response.

27.4.1.1 Reads

Read transfers of any size can be configured to complete with a zero or one additional wait state response on the AHB.

27.4.1.2 Optional read wait-state

The RAM 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

Note: The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the RAM controller clock frequency. If system RAM is running at the same frequency as the RAM 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 PRAM controller clock frequency.

27.4.1.3 Writes

Internal SRAM write operations are performed on the following byte boundaries:

- 1 byte (0:7 bits)
- 2 bytes (0:15 bits)
- 4 bytes or 1 word (0:31 bits)

If the entire 32 data bits are written to SRAM, no read operation is performed and the ECC is calculated across the 32-bit data bus. The 7-bit ECC is appended to the data segment and written to SRAM.

If the write operation is less than the entire 32-bit data width (1 or 2-byte segment), the following occurs:

1. The ECC mechanism checks the entire 32-bit data bus for errors, detecting and either correcting or flagging errors.
2. The write data bytes (1 or 2-byte segment) are merged with the corrected 32 bits on the data bus.
3. The ECC is then calculated on the resulting 32 bits formed in the previous step.
4. The 7-bit ECC result is appended to the 32 bits from the data bus, and the 39-bit value is then written to SRAM.

27.4.1.4 Unaligned writes

The RAM controller is compliant with the AMBA-AHB2.v6 Extensions specification with regard to 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.

It should be noted that unaligned writes always generate read-modify-write operations in the RAM controller in order to preserve the validity of the ECC codeword.

27.4.2 Initialization/application information

To use the SRAM, the ECC must check all bits that require initialization after power on. All writes must specify an even number of registers performed on 32-bit word-aligned boundaries. If the write is not for the entire 32 bits (8 or 16 bits), a read/modify/write operation is generated that checks the ECC value upon the read.

To initialize the ECC, it is essential that each memory address is written to a known value before it is read. 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. Without writing an address to a known value first, a read from this address will most likely generate an uncorrectable ECC event.

It is possible to initialize SRAM memory space via a stored 32-bit word instruction such as Store Multiple Word (e_stmw) in Power Architecture VLE.

28 Flash memory controller (PFLASH Controller)

28.1 Introduction

The flash memory controller has two functions:

- It acts as an interface between the system bus (AHB-Lite 2.v6) and the flash array.
- It serves as the interface to the on-chip overlay RAM and off-chip buddy device.[e]

The flash memory controller supports one 32-bit AHB buses and a 128-bit read data interface to the flash memory array. Each AHB port contains a 4-entry, 2-way set-associative mini-cache as well as an associated controller that prefetches sequential lines of data from the flash arrays into the mini-cache. This buffer mechanism serves to deliver flash read data with zero-wait state response on lines that reside in the cache. AHB requests that miss the cache generate the needed flash array access and are forwarded to the AHB upon completion. Each mini-cache entry is 128 bits in size, matching the code flash array page size and providing 128 bytes of high-speed local storage.

28.2 Features

The following list summarizes the key features of the flash memory controller:

- One 32-bit AHB interface ports (p0).
- 128-bit read data bus + 64-bit write data bus
- Configurable read buffering and line prefetching support via 4-entry, 2-way set-associative mini-cache plus prefetch controller per AHB port to provide single-cycle “buffer hit” read response
- Configurable access control based on read/write and AHB master ID attributes
- Configurable access timing (wait-state programmable) allowing use in a wide range of frequency targets
- Support for reporting of single- and multi-bit flash ECC events on a 64-bit doubleword boundary
- Configurable overlay remapping of logical flash accesses to on-chip calibration RAM, extended off-chip calibration RAM or on-chip system RAM
- Nexus trace stream interface support for message dumping to the on-chip overlay RAM

28.3 Block diagrams

Figure 237 provides a block diagram showing the flash memory controller and the attached flash array.

Note: This module is also known as the Platform Flash controller, that is, PFlash or PFLASH_MP55.

e. The term “buddy device” refers to an optional companion chip with additional memory and debug features that is paired with a production chip during system development. It is not used in production devices.

Figure 237. Platform-centric block diagram with PFLASH_MP55 flash memory controller

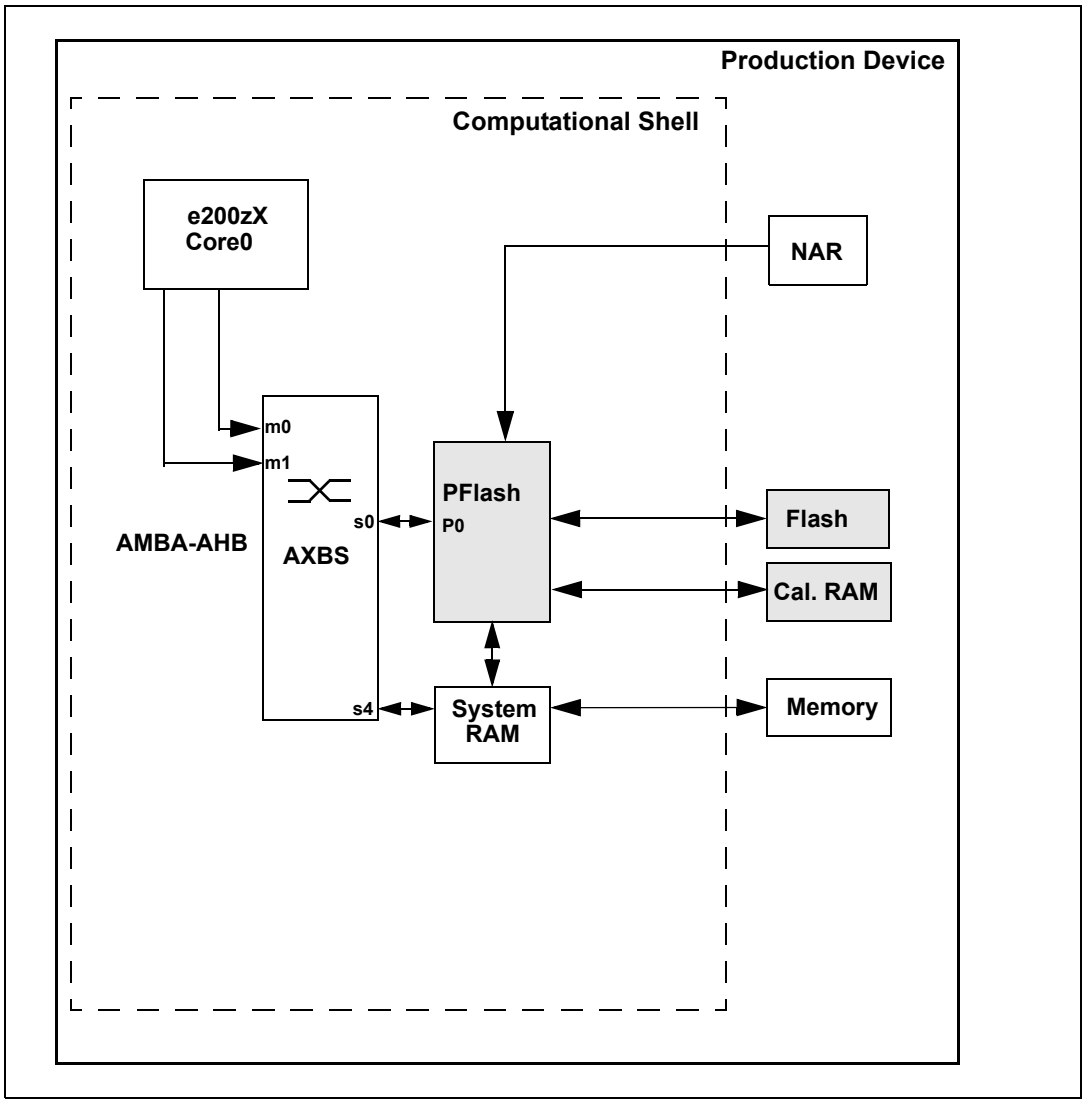
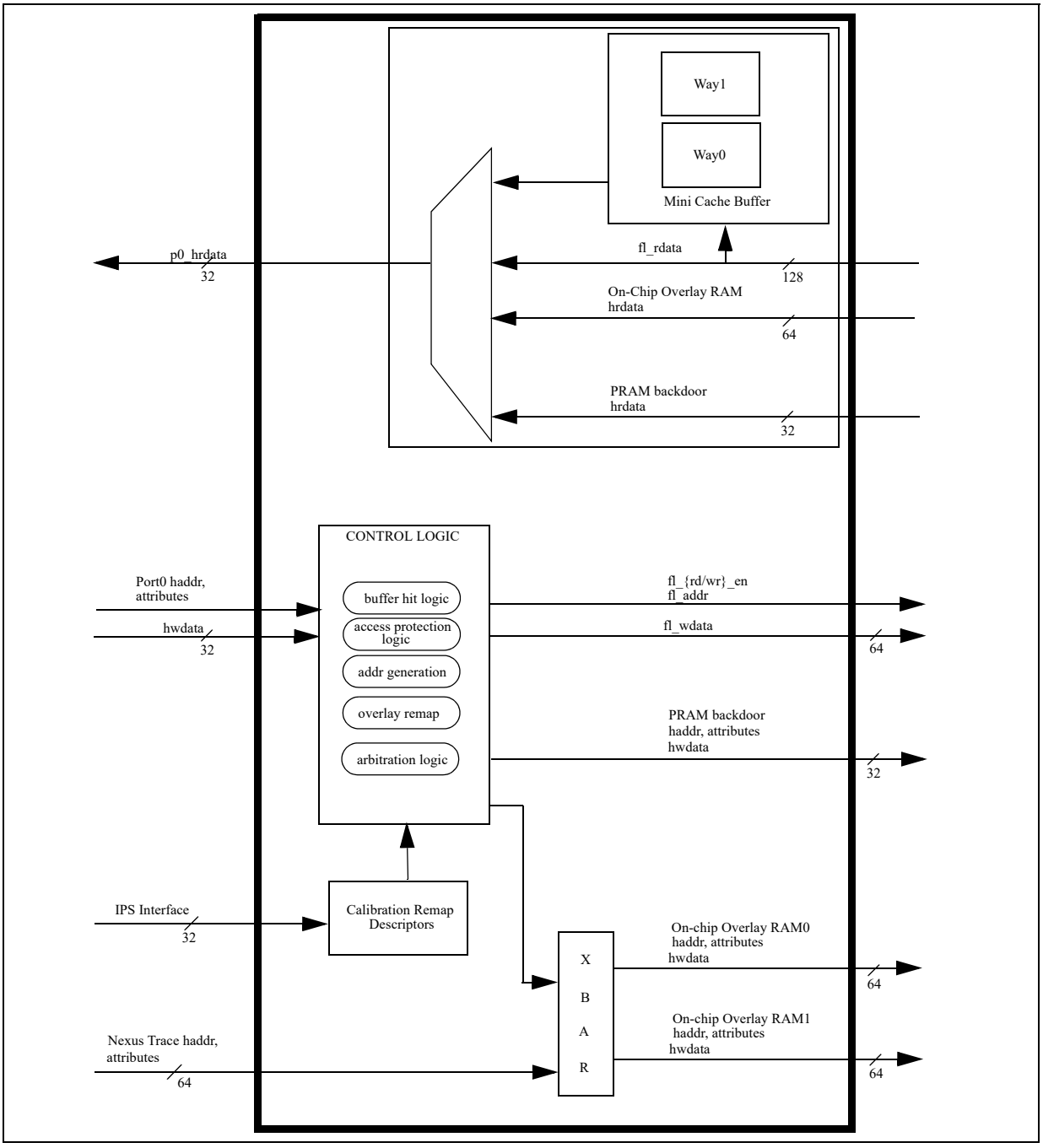


Figure 238. PFLASH_MP55 block diagram



28.4 Flash memory controller memory map

28.4.1 Overview

The flash memory controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model is partitioned into two groups: flash access configuration and overlay remapping 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 midst of an operation will result in non-deterministic behavior.

Table 282. Flash memory controller memory map

Address offset	Register	Access	Reset value	Location
0x000	Platform Flash Configuration Register 1 (PFCR1)	R/W	0x0000_0600	*on page 511*
0x008	Platform Flash Configuration Register 3 (PFCR3)	R/W	0x0000_0000	*on page 515*
0x00c	Platform Flash Access Protection Register (PFAPR)	R/W	0x1111_1111	*on page 516*
0x010	Platform Flash Calibration Remap Control Register (PFCRCR)	R/W	0x0000_0000	*on page 520*
0x014	Platform Flash Calibration Remap Descriptor Enable Register (PFCRDE)	R/W	0x0000_0000	*on page 520*
0x100	Platform Flash Calibration Region Descriptor 0 (PFCRD0)	R/W	0x0000_0000	*on page 527*
0x110	Platform Flash Calibration Region Descriptor 1 (PFCRD1)	R/W	0x0000_0000	*on page 527*
0x120	Platform Flash Calibration Region Descriptor 2 (PFCRD2)	R/W	0x0000_0000	*on page 527*
0x130	Platform Flash Calibration Region Descriptor 3 (PFCRD3)	R/W	0x0000_0000	*on page 527*
0x140	Platform Flash Calibration Region Descriptor 4 (PFCRD4)	R/W	0x0000_0000	*on page 527*
0x150	Platform Flash Calibration Region Descriptor 5 (PFCRD5)	R/W	0x0000_0000	*on page 527*
0x160	Platform Flash Calibration Region Descriptor 6 (PFCRD6)	R/W	0x0000_0000	*on page 527*
0x170	Platform Flash Calibration Region Descriptor 7 (PFCRD7)	R/W	0x0000_0000	*on page 527*
0x180	Platform Flash Calibration Region Descriptor 8 (PFCRD8)	R/W	0x0000_0000	*on page 527*
0x190	Platform Flash Calibration Region Descriptor 9 (PFCRD9)	R/W	0x0000_0000	*on page 527*
0x1A0	Platform Flash Calibration Region Descriptor 10 (PFCRD10)	R/W	0x0000_0000	*on page 527*
0x1B0	Platform Flash Calibration Region Descriptor 11 (PFCRD11)	R/W	0x0000_0000	*on page 527*
0x1C0	Platform Flash Calibration Region Descriptor 12 (PFCRD12)	R/W	0x0000_0000	*on page 527*
0x1D0	Platform Flash Calibration Region Descriptor 13 (PFCRD13)	R/W	0x0000_0000	*on page 527*
0x1E0	Platform Flash Calibration Region Descriptor 14 (PFCRD14)	R/W	0x0000_0000	*on page 527*
0x1F0	Platform Flash Calibration Region Descriptor 15 (PFCRD15)	R/W	0x0000_0000	*on page 527*

Table 282. Flash memory controller memory map(Continued)

Address offset	Register	Access	Reset value	Location
0x200	Platform Flash Calibration Region Descriptor 16 (PFCRD16)	R/W	0x0000_0000	on page 527
0x210	Platform Flash Calibration Region Descriptor 17 (PFCRD17)	R/W	0x0000_0000	on page 527
0x220	Platform Flash Calibration Region Descriptor 18 (PFCRD18)	R/W	0x0000_0000	on page 527
0x230	Platform Flash Calibration Region Descriptor 19 (PFCRD19)	R/W	0x0000_0000	on page 527
0x240	Platform Flash Calibration Region Descriptor 20 (PFCRD20)	R/W	0x0000_0000	on page 527
0x250	Platform Flash Calibration Region Descriptor 21 (PFCRD21)	R/W	0x0000_0000	on page 527
0x260	Platform Flash Calibration Region Descriptor 22 (PFCRD22)	R/W	0x0000_0000	on page 527
0x270	Platform Flash Calibration Region Descriptor 23 (PFCRD23)	R/W	0x0000_0000	on page 527
0x280	Platform Flash Calibration Region Descriptor 24 (PFCRD24)	R/W	0x0000_0000	on page 527
0x290	Platform Flash Calibration Region Descriptor 25 (PFCRD25)	R/W	0x0000_0000	on page 527
0x2A0	Platform Flash Calibration Region Descriptor 26 (PFCRD26)	R/W	0x0000_0000	on page 527
0x2B0	Platform Flash Calibration Region Descriptor 27 (PFCRD27)	R/W	0x0000_0000	on page 527
0x2C0	Platform Flash Calibration Region Descriptor 28 (PFCRD28)	R/W	0x0000_0000	on page 527
0x2D0	Platform Flash Calibration Region Descriptor 29 (PFCRD29)	R/W	0x0000_0000	on page 527
0x2E0	Platform Flash Calibration Region Descriptor 30 (PFCRD30)	R/W	0x0000_0000	on page 527
0x2F0	Platform Flash Calibration Region Descriptor 31 (PFCRD31)	R/W	0x0000_0000	on page 527

28.4.1.1 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_M9 PFE	P0_M8 PFE	P0_M7 PFE	P0_M6 PFE	P0_M5PFE	P0_M4 PFE	P0_M3 PFE	P0_M2 PFE	P0_M1 PFE	P0_M0 PFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	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_IFPFEN	0	P0_PFLIM		P0_BFEN
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 239. PFCR1 register

Table 283. 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

Table 283. PFCR1 field descriptions(Continued)

Field	Description
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
01 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
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

Table 283. PFCR1 field descriptions(Continued)

Field	Description
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. Shown below are the maximum operating frequencies for legal RWSC settings based on specified access times at 150C. The integrator is strongly encourage to verify these settings based on actual silicon results. RWSC=0: 0MHz < fsys < 29.4MHz RWSC=1: 29.4MHz < fsys < 58.8MHz RWSC=2: 58.8MHz < fsys < 88.2MHz RWSC=3: 88.2MHz < fsys < 117.6MHz RWSC=4: 117.6MHz < fsys < 147.0MHz RWSC=5: 147.0MHz < fsys < 176.5MHz RWSC=6: 176.5MHz < fsys < 205.9MHz RWSC=7: 205.9MHz < fsys < 235.5MHz This field is set to 5’b00110 by hardware reset. 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 283. 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. 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 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 is cleared by 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.

28.4.1.2 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		0	0	0	0	0	0	0	0	0	0	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	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 241. PFCR3 register

Table 284. PFCR3 field descriptions

Name	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
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 both, 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.

28.4.1.3 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 242. PFAPR register

Table 285. PFAPR field descriptions

Name	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

Table 285. PFAPR field descriptions(Continued)

Name	Description
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
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

Table 285. PFAPR field descriptions(Continued)

Name	Description
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
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

28.4.1.4 PFlash Calibration Remap Control Register (PFCRCR)

The PFCRCR is used to globally enable/disable the calibration remap function.

Offset 0x010

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	SAFE_CAL	0	0	0	IRMEN	0	0	0	GRMEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 243. PFCRCR register

Table 286. PFCRCR field descriptions

Field	Description
23 SAFE_CAL	Safe Calibration. This bit facilitates the use of safety-critical calibration data. When this bit is enabled, the flash memory controller is configured to perform redundancy checking on the calibration remap function and reduce the total number of potential calibration regions from 32 to 16. This field is initialized by hardware reset. 0 Established calibration overlay regions are not considered safety-critical. 1 Established calibration overlay regions are considered safety-critical.
27 IRMEN	Instruction Remap. This bit enables calibration remapping evaluation on instruction fetches. When PFCRCR[GRMEN] is disabled, this bit is ignored. This field is initialized by hardware reset. 0 Calibration remap evaluation is performed on any incoming data fetch requests only. 1 Calibration remap evaluation is performed on all incoming flash access requests - data and instruction fetches.
31 GRMEN	Global Remap Enable. This bit globally enables or disables the calibration remapping evaluation on system flash accesses. This field is initialized by hardware reset. 0 Calibration remap evaluation is not performed on any incoming system flash access requests. 1 Calibration remap evaluation is performed on all incoming system flash access requests.

28.4.1.5 PFlash Calibration Remap Descriptor Enable Register (PFCRDE)

The PFlash Calibration Remap Descriptor Enable Register (PFCRDE) is used to enable or disable up to 32 calibration remap descriptors. Note that there is also a global remap enable

(PFCRCR[GRMEN]) that also must be asserted in conjunction with the individual CRDnEN flags to enable a given remap descriptor.

Offset 0x014

Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRD0EN	CRD1EN	CRD2EN	CRD3EN	CRD4EN	CRD5EN	CRD6EN	CRD7EN	CRD8EN	CRD9EN	CRD10EN	CRD11EN	CRD12EN	CRD13EN	CRD14EN	CRD15EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRD16EN	CRD17EN	CRD18EN	CRD19EN	CRD20EN	CRD21EN	CRD22EN	CRD23EN	CRD24EN	CRD25EN	CRD26EN	CRD27EN	CRD28EN	CRD29EN	CRD30EN	CRD31EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 244. PFCRDE register

Table 287. PFCRDE field descriptions

Field	Description
0 CRD0EN	Calibration Remap Descriptor 0 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
1 CRD1EN	Calibration Remap Descriptor 1 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
2 CRD2EN	Calibration Remap Descriptor 2 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

Table 287. PFCRDE field descriptions(Continued)

Field	Description
3 CRD3EN	Calibration Remap Descriptor 3 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
4 CRD4EN	Calibration Remap Descriptor 4 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
5 CRD5EN	Calibration Remap Descriptor 5 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
6 CRD6EN	Calibration Remap Descriptor 6 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
7 CRD7EN	Calibration Remap Descriptor 7 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

Table 287. PFCRDE field descriptions(Continued)

Field	Description
8 CRD8EN	Calibration Remap Descriptor 8 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
9 CRD9EN	Calibration Remap Descriptor 9 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
10 CRD10EN	Calibration Remap Descriptor 10 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
11 CRD11EN	Calibration Remap Descriptor 11 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
12 CRD12EN	Calibration Remap Descriptor 12 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

Table 287. PFCRDE field descriptions(Continued)

Field	Description
13 CRD13EN	Calibration Remap Descriptor 13 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
14 CRD14EN	Calibration Remap Descriptor 14 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
15 CRD15EN	Calibration Remap Descriptor 15 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
16 CRD16EN	Calibration Remap Descriptor 16 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
17 CRD17EN	Calibration Remap Descriptor 17 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

Table 287. PFCRDE field descriptions(Continued)

Field	Description
18 CRD18EN	Calibration Remap Descriptor 18 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
19 CRD19EN	Calibration Remap Descriptor 19 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
20 CRD20EN	Calibration Remap Descriptor 20 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
21 CRD21EN	Calibration Remap Descriptor 21 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
22 CRD22EN	Calibration Remap Descriptor 22 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

Table 287. PFCRDE field descriptions(Continued)

Field	Description
23 CRD23EN	Calibration Remap Descriptor 23 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
24 CRD24EN	Calibration Remap Descriptor 24 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
25 CRD25EN	Calibration Remap Descriptor 25 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
26 CRD26EN	Calibration Remap Descriptor 26 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
27 CRD27EN	Calibration Remap Descriptor 27 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

Table 287. PFCRDE field descriptions(Continued)

Field	Description
28 CRD28EN	Calibration Remap Descriptor 28 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
29 CRD29EN	Calibration Remap Descriptor 29 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
30 CRD30EN	Calibration Remap Descriptor 30 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
31 CRD31EN	Calibration Remap Descriptor 31 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

28.4.1.6 PFlash Calibration Region Descriptor *n* (PFCRD*n*)

Each 96-bit (12 byte) region descriptor specifies an overlay region where a flash access can be remapped during calibration and debug. The calibration remap descriptors are organized sequentially as 128-bit (16 byte) structures in the Platform Flash Controller's programming model. Each of the three 32-bit words that define a single calibration region are detailed in the subsequent sections; the fourth word is unused.

28.4.1.6.1 PFlash Calibration Region Descriptor *n*, Word0 (PFCRD*n*.Word0)

The first word of the flash memory controller overlay region descriptor defines the 0-modulo-*size* logical start (byte) address of the calibration remap region. It is software's responsibility

to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

Successful writes to this word clear the calibration remap descriptor's valid bit.

Offset 0x100 + (16*n) + 0x00

Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	LSTARTADDR															
Reset	0	0	0	0	0	0	0	0	0	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	LSTARTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 245. PFlash Calibration Region Descriptor *n*, Word0 (PFCRD*n*.Word0)

Table 288. PFlash Calibration Region Descriptor *n*, Word 0 description

Name	Description
0–27 LSTARTADDR	Logical Start Address. This field defines the most significant bits of the 0-modulo-size logical start address of the overlay remap region. It corresponds to the logical system address which maps to the flash memory space. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.

28.4.1.6.2 PFlash Calibration Region Descriptor *n*, Word1 (PFCRD*n*.Word1)

The second word of the flash memory controller calibration region descriptor defines the 0-modulo-*size* byte physical start (byte) address of the calibration remap region. The contents of this word define the targeted destination overlay memory. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

Successful writes to this word clear the calibration remap descriptor's valid bit.

Offset 0x100 + (16*n) + 0x04

Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSTARTADDR															
Reset	0	0	0	0	0	0	0	0	0	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	PSTARTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 246. PFlash Calibration Region Descriptor *n*, Word1 (PFCRD*n*.Word1)

Table 289. PFlash Overlay Region Descriptor *n*, Word 1 description

Name	Description
0–27 PSTARTADDR	Calibration Remap Descriptor *n* Physical Start Address - This field defines the most significant bits of the 0-modulo-size physical start byte address of the calibration remap descriptor. This address corresponds to the physical address which maps to the destination calibration overlay memory. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.

28.4.1.6.3 PFlash Calibration Region Descriptor *n*, Word2 (PFCRD*n*.Word2)

The third word of the calibration region descriptor defines a per-master calibration remap enable and the remap region size. For cacheable spaces being remapped, the minimum region size is 32 bytes to match the flash page and cache line sizes. Writes to this word clear the calibration remap descriptor's valid bit.

Offset 0x100 + (16*n) + 0x08

Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	M0EN	M1EN	M2EN	M3EN	M4EN	M5EN	M6EN	M7EN	M8EN	M9EN	M10EN	M11EN	M12EN	M13EN	M14EN	M15EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R												CRDSize				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 247. PFlash Calibration Region Descriptor *n*, Word2 (PFCRDn.Word2)

Table 290. PFlash Overlay Region Descriptor *n*, Word 2 description

Name	Description
0-15 MxEN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and data is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and data is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master X as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master X as defined by this region descriptor.
4–0 CRDSize	Calibration Remap Descriptor *n* Size - This field specifies the size of the calibration remap region. 00000 Region size is 1 byte. 00001 Region size is 2 bytes. Low-order bit of LSTARTADDR and PSTARTADDR must be zero 00010 Region size is 4 bytes. Low-order 2 bits of LSTARTADDR and PSTARTADDR must be zero 00011 Region size is 8 bytes. Low-order 3 bits of LSTARTADDR and PSTARTADDR must be zero 00100 Region size is 16 bytes. Low-order 4 bits of LSTARTADDR and PSTARTADDR must be zero 00101 Region size is 32 bytes. Low-order 5 bits of LSTARTADDR and PSTARTADDR must be zero 00110 Region size is 64 bytes. Low-order 6 bits of LSTARTADDR and PSTARTADDR must be zero 00111 Region size is 128 bytes. Low-order 7 bits of LSTARTADDR and PSTARTADDR must be zero 01000 Region size is 256 bytes. Low-order 8 bits of LSTARTADDR and PSTARTADDR must be zero 01001 Region size is 512 bytes. Low-order 9 bits of LSTARTADDR and PSTARTADDR must be zero 01010 Region size is 1 KB. Low-order 10 bits of LSTARTADDR and PSTARTADDR must be zero 01011 Region size is 2 KB. Low-order 11 bits of LSTARTADDR and PSTARTADDR must be zero 01100 Region size is 4 KB. Low-order 12 bits of LSTARTADDR and PSTARTADDR must be zero 01101 Region size is 8 KB. Low-order 13 bits of LSTARTADDR and PSTARTADDR must be zero 01110 Region size is 16 KB. Low-order 14 bits of LSTARTADDR and PSTARTADDR must be zero 01111 Region size is 32 KB. Low-order 15 bits of LSTARTADDR and PSTARTADDR must be zero 10000 Region size is 64 KB. Low-order 16 bits of LSTARTADDR and PSTARTADDR must be zero 10001 Region size is 128 KB. Low-order 17 bits of LSTARTADDR and PSTARTADDR must be zero 10010 Region size is 256 KB. Low-order 18 bits of LSTARTADDR and PSTARTADDR must be zero 10011 Region size is 512 KB. Low-order 19 bits of LSTARTADDR and PSTARTADDR must be zero 10100 Region size is 1 MB. Low-order 20 bits of LSTARTADDR and PSTARTADDR must be zero 10101 Region size is 2 MB. Low-order 21 bits of LSTARTADDR and PSTARTADDR must be zero 10110 Region size is 4 MB. Low-order 22 bits of LSTARTADDR and PSTARTADDR must be zero 10111 Region size is 8 MB. Low-order 23 bits of LSTARTADDR and PSTARTADDR must be zero 11000 Reserved 11001 Reserved 11010 Reserved 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 Reserved

28.5 Functional description

As shown in *Figure 237* the flash memory controller interfaces between the AHB system bus port, Nexus trace, flash memory array, overlay RAM, system RAM, and off-chip Buddy Device.

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 two ways of the flash memory controller mini-cache. Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

28.5.1 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.

28.5.2 Access protections

28.5.2.1 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 28.4.1.3: 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.

28.5.2.2 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].

28.5.3 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 globally enabled, the calibration remap evaluation is also performed during the first AHB data phase cycle. If the access is to be remapped, the appropriate access is routed to the destination memory during the second 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.

28.5.4 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.

28.5.5 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 28.4.1.3, 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 28.4.1.2, 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 28.5.6, Flash error response operation*).

28.5.6 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 flash array documentation.

28.5.7 Censorship

The entire flash space is defined by three major regions:

- Code flash (instruction and constant data)
- Data flash (data for EEPROM emulation)
- TEST flash

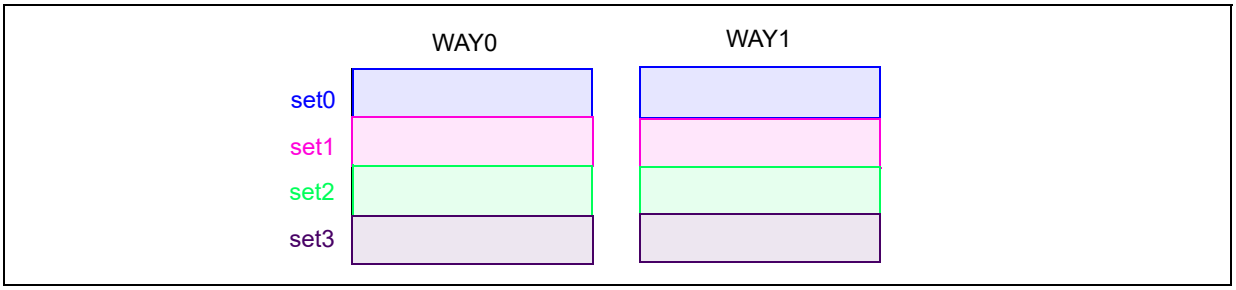
Each section can be independently censored, and the flash memory controller provides independent censorship control inputs for each region on a read vs. write basis.

28.5.8 Line read buffers and prefetch operation

The AHB port of the flash memory controller contains a two-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the flash arrays. Each 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 28.4.1.1: 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 248. 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 28.4.1.1, 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, PFCR1[P0_BFEN] must be set to '1', PFCR1[P0_PFLIM] must be non-zero, and either PFCR1[P0_IPFEN] or PFCR1[P0_DPFEN] must be asserted. Refer to *Section 28.4.1.1, Platform Flash Configuration Register 1 (PFCR1)* for a description of these controls.

28.5.9 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 28.4.1.1, Platform Flash Configuration Register 1 (PFCR1)* for a description of these controls. Prefetches are never triggered by write cycles.

28.5.10 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to *Section 28.4.1.1, Platform Flash Configuration Register 1 (PFCR1)* for a description of these controls.

28.5.11 Buffer allocation

Allocation of the line read buffers is controlled via the PFCR3 control register, specifically the line buffer configuration (P0_WCFG) field. Refer to *Section 28.4.1.2, 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.

28.5.12 PFlash calibration remap support

The flash memory controller supports calibration development by providing a remapping function to route flash accesses to overlay RAM. Recall this family of devices includes a comprehensive set of features including dedicated overlay RAM arrays to aid calibration and debug. The overlay function supports the following features:

- Can be mapped over internal flash memory
  - Allows calibration of constant data without requirement for additional external RAMs and calibration memory interfaces
- Can be used to hold device debug trace stream
  - Allows limited trace based debug without requirement for complex debug tool hardware and access to high speed trace port
- Protected with error detection and correction mechanism
- Accesses to all overlay RAMs can have the same timing as accesses to internal flash
  - Timing for calibration accesses to a particular overlay RAM is only guaranteed if it is not being used as a destination for trace streaming at the same time
- Support for up to 32 distinct calibration remap regions.

The calibration remap function supports up to two different overlay RAM targets:

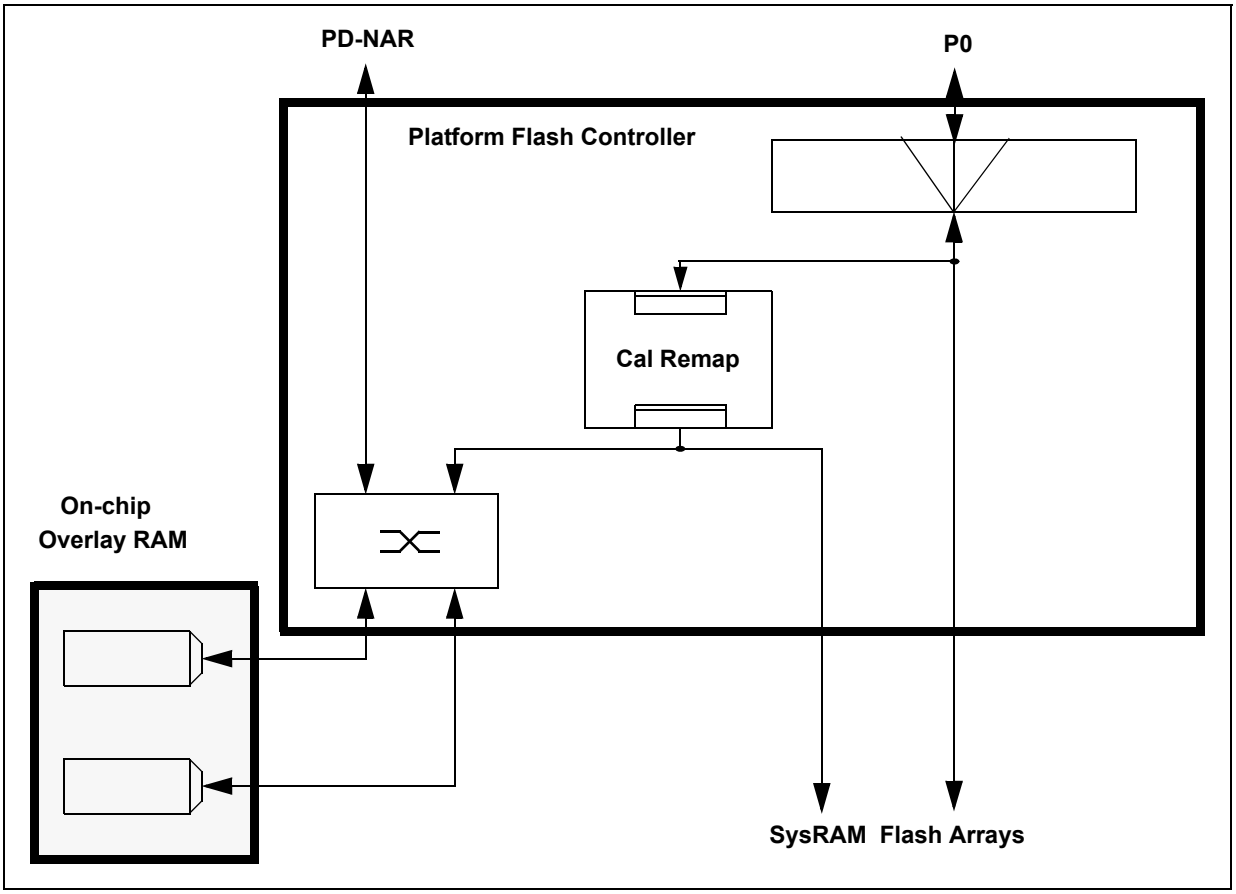
1. An internal overlay RAM is included on all standard production devices.
2. A portion of the system RAM can be used for overlay.

where the *overlay RAM is selected based on the translated physical address*.

To better understand the functionality of the calibration remap, consider more detailed block diagrams, one showing the overall structure of the calibration remap function and another showing more detail on the actual remap logic. Recall the flash memory controller implements a hardware interconnection matrix connecting the four input ports and four destination memories. See *Figure 249*.

Figure 249 illustrates the calibration remap connection matrix.

Figure 249. PFlash calibration remap connection matrix



Note that the 64-bit on-chip overlay RAM is split into 2 equal-sized memories to allow concurrent accesses for calibration data from the P0 port plus trace data writes from the Production Device Nexus Aurora Router (PD-NAR).

Note that when a reset event asserts while an access to the on-chip overlay RAM is requested, the flash controller aborts the array access.

The translated physical address defines the destination memory. *Table 291* shows the base addresses for the calibration data memories as defined by the system memory map.

Table 291. Calibration data memory base addresses

Calibration memory	Base address	Possible access sources
Flash Arrays	0x0{8,9,A}--_----	P0, P1
System RAM	0x40{0,1}-_----	P0, P1

Table 291. Calibration data memory base addresses(Continued)

Calibration memory	Base address	Possible access sources
On-chip Overlay RAM	0x0D00_----	P0, P1, PD-NAR
Buddy Device (Extended) Overlay RAM	0x0C{0,1}-_----	P0, P1

Note that the trace data from the Nexus Aurora Router (PD-NAR) can only be routed to the on-chip overlay RAM; there is a different connection path in the Buddy Device from the BD-NAR into the extended overlay RAM.

The system RAM connection from the calibration remap logic is implemented via *a private 32-bit connection to the platform's RAM controller*. Since the calibration access is defined to have the same access time as a normal flash array access, remapped references to the system RAM are treated as the highest priority request by the platform's RAM controller.

Accesses to the flash arrays always use the logical, non-remapped system address. Calibration data store references to the system RAM, on-chip or buddy device (extended) overlay RAM always use the physical, non-remapped system address. Conversely, calibration data load references can be accessed using a *translated physical address or, if provided by the requesting master, the physical, non-remapped system address.* Doubleword-sized data store references to the on-chip overlay RAM are supported. Byte-, halfword-, word- and doubleword-sized store references to the on-chip calibration RAM or the buddy device (extended) overlay RAM are supported. The read data interface of system RAM is 32-bit wide and the on-chip overlay RAM interfaces are each 64-bit wide, whereas the flash read data interface is 128-bit wide. As a result, a remapped calibration access initiates a four-beat burst sequence to mimic the throughput of a single flash access. There is a direct relationship between the operating frequency range and the required number of wait states to correctly sample flash read data. By contrast, a remapped calibration access has a fixed 4-beat latency. The access time of each individual beat depends on the overlay target (system RAM, on-chip overlay RAM, or buddy device). Therefore, PFCR1[RWSC] must be programmed to a value such that the overall flash access latency sufficiently covers the required access time to complete a 4-beat burst sequence from the slowest available overlay target at the specified device operating frequency to guarantee the latency for an initial, random calibration access accurately mimics the flash access time. Expressed mathematically:

$$(\mathrm{RWSC} + 1) \geq ((\mathrm{OverlayRAMTranferLatency} \times 4) + 1)$$

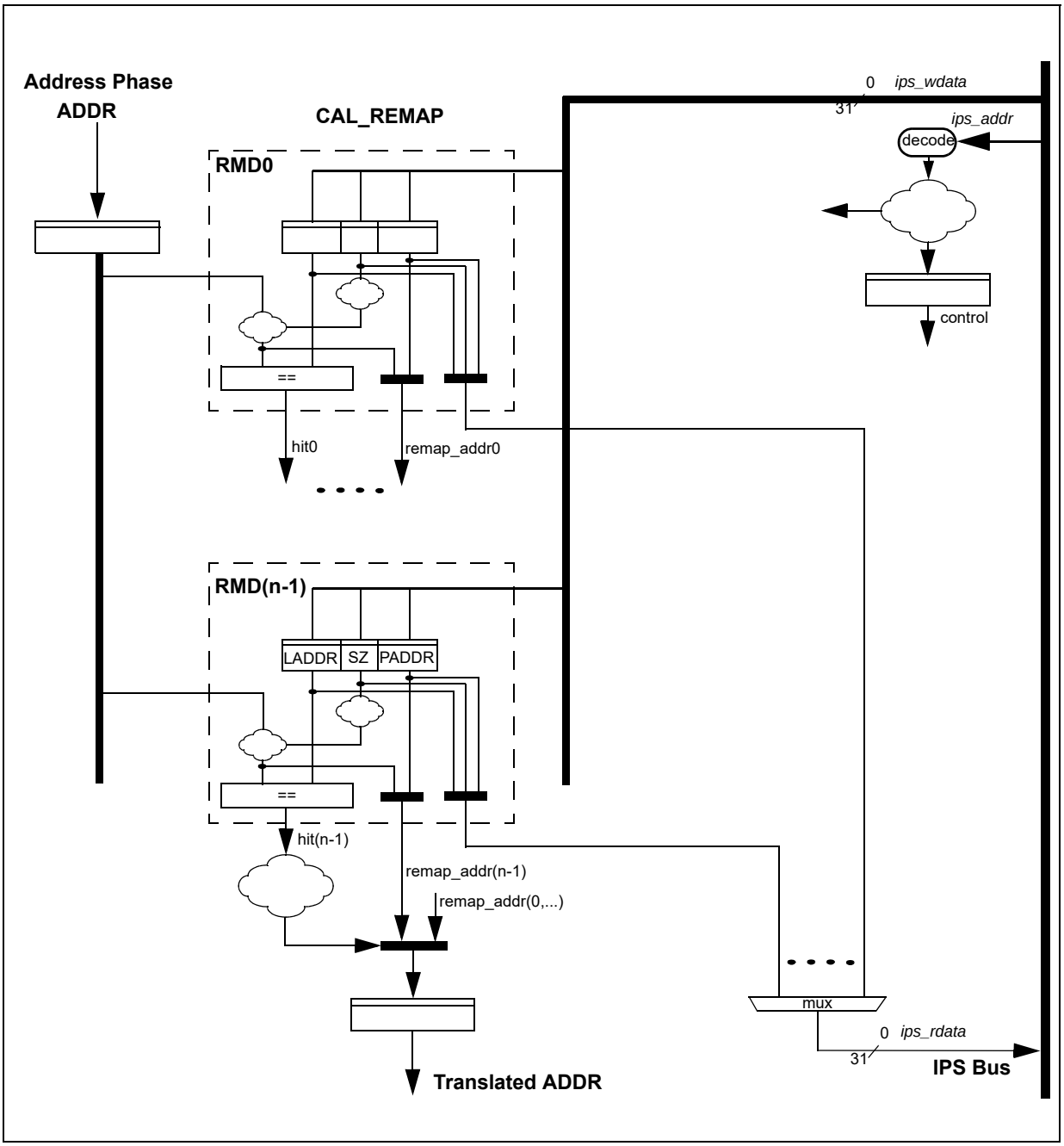
$$\text{where} \quad \mathrm{RWSC} = \mathrm{ceiling}\left(\frac{\mathrm{IntrinsicFlashAccessTime}}{\mathrm{OperatingFrequencyClockPeriod}}\right)$$

Notice the overall latency as perceived by the requesting master on a random, initial flash access incurs two additional cycles beyond the intrinsic flash access time. The overall latency on a remapped overlay access incurs one additional cycle beyond the burst access time. These cycles accounts for pipeline stalls incurred by the flash controller state machine. Calibration data that is available in the flash controller's mini-cache is returned with single-cycle latency

There are memory-mapped calibration remap (region) descriptor (CRD*n*) registers which define the required address translation. Each calibration remap descriptor includes the

logical base address, the translated physical base address and a region size (plus other control bits). The calibration remap evaluation is initiated when the PFLASH_MP55 flash controller is presented with a flash access request from any system bus master to a location in the mirrored flash address space. Conversely, references using the associated non-mirrored flash system address are not evaluated for calibration remapping. The address translation mechanism requires the logical and physical base addresses of the region be aligned on 0-modulo-size boundaries. The translation function is performed during the first AHB data phase cycle. See *Figure 250*.

Figure 250. PFlash calibration remap logic detail



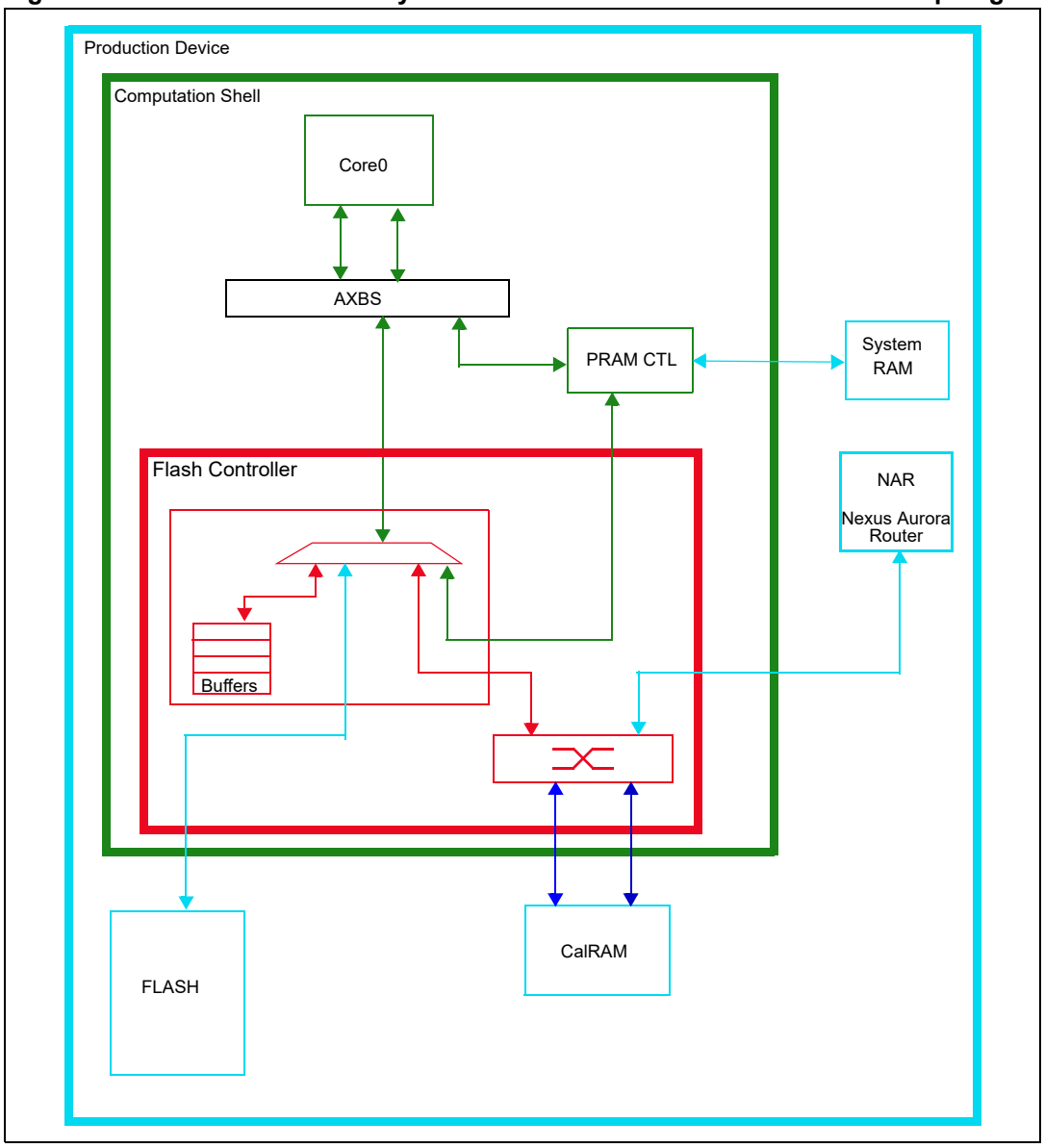
Each instantiation of the basic calibration remap function {RMD0,..., RMD(*n*-1)} in *Figure 250* includes the three descriptor registers containing the logical start address (LADDR in the figure), the physical start address (PADDR) and the region size (SZ). The region size is decoded to create the appropriate address bit enables which are then applied to the registered logical flash address. An equality comparator then compares the adjusted logical flash address against the logical address from the calibration remap descriptor to determine the region “hit”. The hit determination is further qualified based on the bus master number that initiated the flash access. At the same time, the logical flash address and the physical address defined in the calibration remap descriptor are merged to form the translated address to be used in the event of a calibration region hit. Aside from the individual instantiations of the calibration remap descriptors, this logic examines all the descriptor hit indicators to select the appropriate translated address.

In the absence of a region “hit” data is returned from the flash at the provided logical address.

It should be noted in the event of *overlapping remap regions*, the calibration remap descriptor with the largest number is used for the address translation, that is, if a logical flash address hits in multiple descriptors, say CRDx and CRDy, the translated address from CRDy is used, given y > x. This definition is selected as it minimizes the gate cost and timing impact of the calibration remap descriptor “hit” logic.

In general, the calibration remap logic operates only on flash *data* accesses. However, flash instruction references can optionally remapped while in debug modes of operation to support the use of software breakpoints when PFCRC[IRMEN] is set.

Figure 251. PFlash Flash memory controller and associated calibration remap targets



28.5.13 Safety considerations

28.5.13.1 ECC Reporting

The flash controller provides ECC event flags for reporting of ECC events which occur on flash and overlay RAM accesses. ECC events are reported when requested; ECC events are not reported on prefetch accesses. The flash controller event flags indicate the occurrence of single-bit correction events, multi-bit error, the relevant system address and associated syndrome value.

28.5.13.1.1 e2eECC and calibration accesses

In addition, the flash memory controller can be configured to treat calibration RAM space as a safety- critical component. By programming PFCRCR[SAFE_CAL] = 1, the flash memory

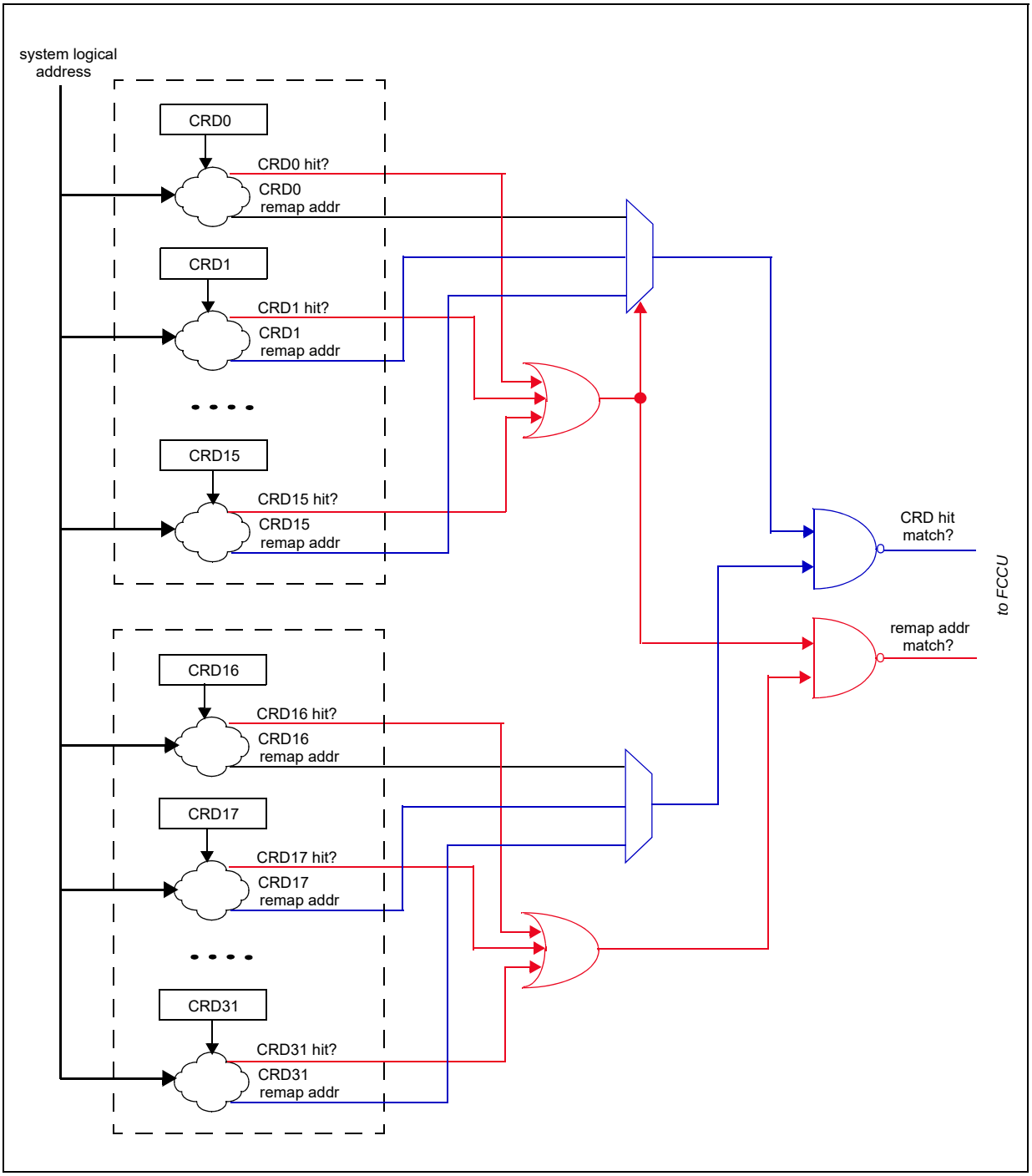
controller treats the 32 available calibration region descriptors (CRD0–CRD31) as a replicated group of 16 region descriptors. More specifically:

- The contents of CRD0 are expected to be redundantly defined in CRD16
- The contents of CRD1 are expected to be redundantly defined in CRD17
- The contents of CRD2 are expected to be redundantly defined in CRD17
- ...and so on
- The contents of CRD15 are expected to be redundantly defined in CRD31

Note: It is the user's responsibility to establish redundant calibration region descriptors by programming the associated pairs of CRDs.

Incoming flash read requests are compared simultaneously against the calibration regions defined in CRD0–15 and the calibration regions defined in CRD16–31. Due to the enabled redundancy, the parallel overlay remap hardware structures should evaluate to symmetric results. If a mismatch is detected, the event is reported to the FCCU and calibration remapping is not performed. Instead, data is returned from the flash.

Figure 252. Safety-critical calibration remap datapath with redundancy



28.5.13.1.2 e2eECC and data flash accesses

ECC events detected on accesses to the data flash blocks are suppressed from being reported to the ECC event collection unit.

In the event of a single-bit correction, the corrected data 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, illegal opcode value is returned to the requesting master as determined by the requesting address, and the non-correctable error event is suppressed from being reported.

28.5.13.2 Flash address generation check

Functional safety coverage of the address and data are handled by ECC performed within the flash and e2eECC performed at the master. Functional safety coverage of the address path and control within the flash memory controller rely on a feedback path between the flash controller and flash. Recall on a requested access to flash, the flash memory controller must decode the system AHB bus signals to generate the corresponding flash interface signals to 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:

fl_enc_addr[23:2]

fl_enc_error

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 each AHB port: p0 and p1.

28.5.13.3 On-Chip Overlay RAM Feedback Check

The integrity of the address and data on an overlay RAM transaction is covered by e2eECC check performed by the requesting master. Functional safety coverage of the address path and control within the flash memory controller is handled by a transaction monitor which verifies the integrity of the transactions between the flash memory controller and the on-chip overlay RAM arrays.The function of the transaction monitor relies on a feedback path between the flash memory controller and the on-chip overlay RAM arrays, 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 flash memory controller transaction monitor to verify the expected transaction.

28.5.14 Array integrity considerations

During an array integrity sequence, the flash array ignores any incoming read requests. The flash provides an output sideband, *fl_aid*, which indicates when an array integrity sequence is in progress. When *fl_aid* is asserted, the flash memory controller aborts any incoming flash access requests and terminates the system bus transfer with an error.

29 Embedded Flash Memory (MP55)

29.1 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.

29.1.1 Overview

The 1584 KB Flash Memory module contains 2 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 one other partition. Each module counts as an RWW partition.

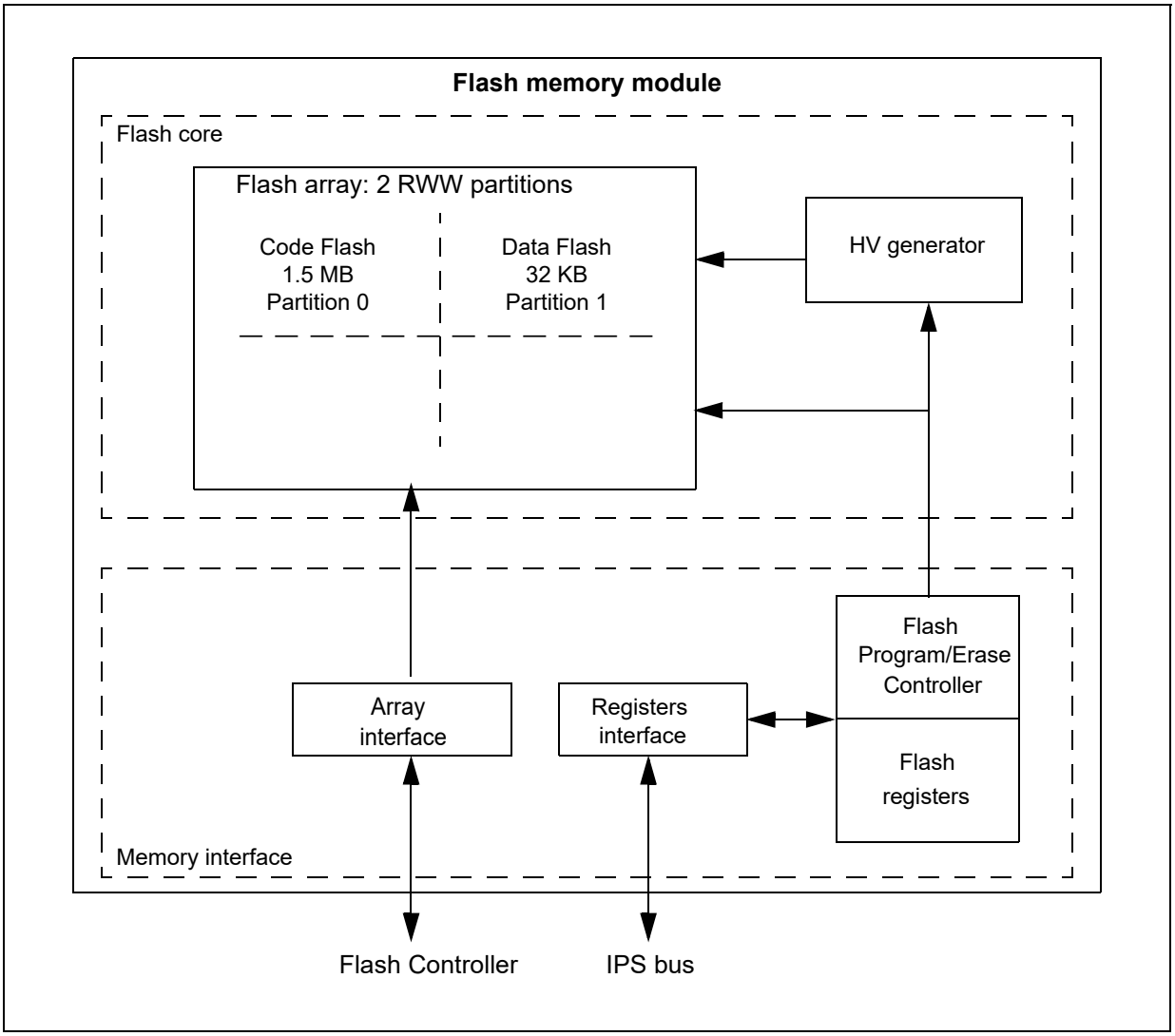
EEPROM emulation is managed in 1 of the modules, referred to as Data Flash.

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 bits 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 2 partitions.

Figure 253. 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 Erase of the flash array requires multiple system clock cycles to complete.

The Program and Erase sequences may be suspended or aborted.

29.1.2 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 °C junction)
- UTEST mode (user-accessible Test modes)
- Critical flops which may affect Flash content are triple-voted for safety application

29.1.3 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 29.4, Functional description*.

29.2 Flash memory map and description

The Flash Memory module general architecture supports total memory sizes ranging from 32 KB to 8 MB of user memory, plus up to 48 KB of Test memory (a portion of which is One Time Programmable by the user).

Optionally, an extra memory space ranging from 32 KB to 512 KB can be used as data space for EEPROM emulation, and can be organized in different RWW partitions (in addition to the main User memory Partition).

There are four User Address Spaces: Low, Mid, High and 256K Address Space:

- Low Address Space must always be present and be up to 256 KB in size and may be organized in a maximum of 14 blocks (16 KB[f], 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[f], 16 KB, 32 KB, and 64 KB blocks).
- 256K Address Space can be present and be up to 7.5 MB in size and may be organized in a maximum of 30 blocks (256 KB).

Note: There is also a special Test Address space which must always be present and be up to 48 KB in size. Test Address space is part of flash Partition 0.

The Flash Memory module can be divided into a minimum of one to a maximum of eight flash partitions:

The total size of each Code flash Partition cannot be greater than 2 MB and the maximum number of blocks cannot exceed 16, including TestFlash. The total size of each Data flash Partition cannot be greater than 256 KB and the max 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 256 KB blocks can be mapped in the 256K address space. 8 KB[f] 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.

f. 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.

29.2.1 Flash array memory map

29.2.1.1 Code flash

Partition 0 accounts for 1.5 MB of the total Flash Memory and is divided in 6 user blocks for code storage. The first 256 KB blocks are organized as 1 × 256 KB. Remaining blocks are 256 KB.

Partition 0 also contains a reserved block named TestFlash in which some One Time Programmable (OTP) user data is stored.

The sectorization of partition 0 of the 1584 KB Flash Memory module is shown in *Table 292*.

Table 292. Flash memory partition 0 memory map

Block	Absolute addresses	Size	Address Space	LOCK	SEL
B0BF	0x0040_4000 to 0x0040_5FFF	8 KB	Low	LOWLOCK[0]	—
B0F1	0x0100_0000 to 0x0103_FFFF	256 KB	256K	A256KLOCK[0]	A256KSEL[0]
B0F2	0x0104_0000 to 0x0107_FFFF	256 KB	256K	A256KLOCK[1]	A256KSEL[1]
B0F3	0x0108_0000 to 0x010B_FFFF	256 KB	256K	A256KLOCK[2]	A256KSEL[2]
B0F4	0x010C_0000 to 0x010F_FFFF	256 KB	256K	A256KLOCK[3]	A256KSEL[3]
B0F5	0x0110_0000 to 0x0113_FFFF	256 KB	256K	A256KLOCK[4]	A256KSEL[4]
B0F6	0x0114_0000 to 0x0117_FFFF	256 KB	256K	A256KLOCK[5]	A256KSEL[5]
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
B0UT	0x0040_0000 to 0x0040_1FFF	8 KB	Low	TSLOCK	—
B0TF	0x0040_8000 to 0x0040_BFFF	16 KB	Reserved	—	—

29.2.1.2 Data flash

Data flash is allocated in partition 1 of the Flash Memory module in 2 × 16 blocks destined for data storage.

The Data Flash sectorization of the Flash Memory module is listed below.

Table 293. Flash Memory Partition 1 memory map

Block	Absolute addresses	Size	Address Space	LOCK	SEL
B1F0	0x0080_0000 to 0x0080_3FFF	16 KB	High	HIGHLOCK[0]	HIGHSEL[0]
B1F1	0x0080_4000 to 0x0080_7FFF	16 KB	High	HIGHLOCK[1]	HIGHSEL[1]
—	—	—	—	—	—

Table 293. Flash Memory Partition 1 memory map(Continued)

Block	Absolute addresses	Size	Address Space	LOCK	SEL
—	—	—	—	—	—
—	—	—	—	—	—

29.2.1.3 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 294*.

Table 294. 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 byte	R/P
0x0008	Reserved	4 byte	R/P
0x000C	Test Mode Disable Seal	4 byte	R/P
0x0010	Test Mode Disable Block Select 0	16 byte	R/P
0x0020	Factory Erase Diary Location	16 byte	R/P
0x0030	Test Mode Disable Block Select 1	16 byte	R/P
0x0040	Customer Single Bit Correction area	32 byte	R
0x0060	Customer Double Bit Detection area	32 byte	R
0x0080	Reserved	32 byte	R/P
0x00A0	Reserved	96 byte	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 byte	R/P
0x0104	Reserved	252 byte	R/P
0x0200	Reserved	3584 byte	R/P
0x1000	Customer OTP data	4096 byte	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 standalone block.

When the UTEST section (via the entire TestFlash block) is protected via Test Mode Disable Seal, the BAF is also protected from any high voltage operation which may alter the content. Access to BAF is read only for FPEC in the event that the test interface is open (Failure Analysis scenario).

29.3 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 case of failing system locations (configurations, redundancy, EmbAlgo firmware), the initialization phase is interrupted and a Fatal Error is flagged.
- In 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.

In this section, the following abbreviations are used:

Table 295. Register abbreviations

Case	Abbreviation	Description
Read/Write	R/W	Software can read and write these bits.
Read/Clear	rc	Software can read and clear these bits.
Read/Clear	w1c	Software can read and clear these bits by writing a one.
Read-only	R	Software can only read these bits.
Write-only	W	Software should only write to these bits.

29.3.1 Register memory maps

This section presents the memory mapping of all the flash memory registers.

29.3.1.1 User registers memory map

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 296*.

Table 296. Flash User Registers memory map

Offset address	Description	Access	Reset value(1)	Location
0x0000	Module Configuration Register (MCR)	R/W(2)	0x0000_0600	*on page 552*
0x0004	Reserved	—	—	—
0x0008	Extended Module Configuration Register (MCRE)	R	0x0601_0000 (2)	*on page 559*
0x000C	Reserved	—	—	—
0x0010	Locking 0 register (LOCK0)	R/W	0xBFFF_FFFF	*on page 562*
0x0014	Locking 1 register (LOCK1)	R/W	0x0000_FFFF	*on page 564*
0x0018	Locking 2 register (LOCK2)	R/W	0x7FFF_FFFF	*on page 565*
0x001C	Reserved	—	—	—
0x0020	Reserved	—	—	—
0x0024	Reserved	—	—	—
0x0028	Reserved	—	—	—
0x002C	Reserved	—	—	—
0x0030	Reserved	—	—	—
0x0034	Reserved	—	—	—
0x0038	Select 0 register (SEL0)	R/W	0x0000_0000	*on page 566*
0x003C	Select 1 register (SEL1)	R/W	0x0000_0000	*on page 567*

Table 296. Flash User Registers memory map(Continued)

Offset address	Description	Access	Reset value(1)	Location
0x0040	Select 2 register (SEL2)	R/W	0x0000_0000	*on page 568*
0x0044	Reserved	—	—	—
0x0048	Reserved	—	—	—
0x004C	Reserved	—	—	—
0x0050	Address Register (ADR)	R	0x0000_0000	*on page 569*
0x0054	User Test 0 register (UT0)	R/W	0x0000_0001	*on page 571*
0x0058	User Multiple Input Signature 0 register (UM0)	R/W	0x0000_0000	*on page 574*
0x005C	User Multiple Input Signature 1 register (UM1)	R/W	0x0000_0000	*on page 574*
0x0060	User Multiple Input Signature 2 register (UM2)	R/W	0x0000_0000	*on page 575*
0x0064	User Multiple Input Signature 3 register (UM3)	R/W	0x0000_0000	*on page 575*
0x0068	Reserved	—	—	—
0x006C	Reserved	—	—	—
0x0070	Reserved	—	—	—
0x0074	Reserved	—	—	—
0x0078	User Multiple Input Signature 8 register (UM8)	R/W	0x0000_0000	*on page 576*
0x007C	User Multiple Input Signature 9 register (UM9)	R/W	0x0000_0000	*on page 576*
0x0080	Over Program Protection 0 (OPP0)	R	0x0000_0000	*on page 577*
0x0084	Over Program Protection 1 (OPP1)	R	0x0000_0000	*on page 578*
0x0088	Over Program Protection 2 (OPP2)	R	0x0000_0000	*on page 579*
0x008C	Reserved	—	—	—
0x0090	Test Mode Disable Password Check (TMD)	R/W	—	*on page 580*

1. Reset value comes from UTEST area within TestFlash, and this assumes the location is erased and all possible blocks are available.
2. Note that R/W register may contain some read only or write only bits.

29.3.2 User register descriptions

This section consists of user register descriptions in address order.

29.3.2.1 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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	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 254. Module Configuration Register (MCR)

Table 297. MCR field descriptions

Field	Description
0–15	Reserved (read only) Writing to this bit has no effect and Reads always return 0.
16 EER	ECC event Error (Read/Clear) 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 (Read/Clear) 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 297. MCR field descriptions(Continued)

Field	Description
18 SBC	Single Bit Correction (Read/Clear) 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 a 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.
19	Reserved (read only) Writing to this bit has no effect and Reads always return 0.
20 PEAS	Program/Erase Access Space (read only). 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 (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 (read only) 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 P/E Abort Latency). – After 0–1 transition of ESUS/PSUS which suspends an Erase/Program operation (within t_{ESUS}/t_{PSUS} time equals to 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 297. MCR field descriptions(Continued)

Field	Description
22 PEG	Program/Erase Good (read only) 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 was 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’. PEG behavior with blocks marked as OPP (though f90_otpen0/1/2 sidebands signal) 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 the occurrence of any error, abort, break point or suspend. The presence of errors can be detected only comparing checksum value stored in UM0–9 and checking MCR flags. 0 Program, Erase operation failed or Program, Erase aborted. 1 Program or Erase operation successful.
23 PECIE	Program/Erase Complete Interrupt Enable (Read/Write). 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 297. MCR field descriptions(Continued)

Field	Description
24 FERS	Factory Erase (Read/Write). FERS is used for Erase and Program operations to enable 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.
25–26	Reserved (read only) Writing to this bit has no effect and Reads always return 0.
27 PGM	Program (Read/Write) 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 be set only under the following conditions: – User Mode Read (ERS is low and UT0[AIE] is low). – Erase Suspend (ERS and ESUS high) with EHV low. PGM can be cleared by the user only when PSUS and EHV are low and DONE is high. PGM is cleared on reset. **Note:** In an erase-suspended program, programming Flash memory locations in blocks which were being operated on in the Erase is forbidden. Command is not accepted and PEG is set to 0 0 Flash is not executing a program sequence. 1 Flash is executing a program sequence.

Table 297. MCR field descriptions(Continued)

Field	Description
28 PSUS	Program Suspend (Read/Write). 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 case of Program Suspend while in Erase Suspend (PSUS=1, ESUS=1), PSUS has to be cleared first in order to resume currently higher level suspended operation that is program. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.
29 ERS	Erase (Read/Write) 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[AIE] 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 297. MCR field descriptions(Continued)

Field	Description
30 ESUS	Erase Suspend (Read/Write) 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.

Table 297. MCR field descriptions(Continued)

Field	Description
31 EHV	Enable High Voltage (Read/Write) 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 298*.

Table 298. 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.

29.3.2.2 Extended Module Configuration Register (MCRE)

The Extended Module Configuration Register value is based on the size, blocks and address space configuration of the Flash Memory module.

Address: 0x0008 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	n8Kh			n256K				n64Kh			n32Kh		n16Kh		
W																
Reset	0	0	0	0	0	1	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	n64Km			n32Km		n16Km			n64Kl			n32Kl		n16Kl		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 255. Extended Module Configuration Register (MCRE)

Table 299. MCRE field descriptions

Field	Description
0	Reserved (read only) Writing to this bit has no effect and Reads always return 0.
1-2 n8Kh	Number of 8 KB blocks in high space. (read only) 00 Zero 8 KB blocks. 01 Two 8 KB blocks. 10 Four 8 KB blocks. 11 Six 8 KB blocks.

Table 299. MCRE field descriptions(Continued)

Field	Description
3–7 n256K	Number of 256K blocks. (read only) 00000 Zero 256K blocks. 00001 One 256K block. 00010 Two 256K blocks. 00011 Three 256K blocks. 00100 Four 256K blocks. 00101 Five 256K blocks. 00110 Six 256K blocks. 00111 Seven 256K blocks 01000 Eight 256K blocks. 01001 Nine 256K blocks. 01010 Ten 256K blocks. ..... 11100 Twenty-Eight 256K blocks (7.0 MB total). 11101 Twenty-Nine 256K blocks (7.25 MB total). 11110 Thirty 256K blocks (7.5 MB total). 11111 Thirty-one 256K blocks (7.75 MB total).
8–10 n64Kh	Number of 64 KB blocks in high space. (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. (read only) 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
13–15 n16Kh	Number of 16 KB blocks in high space. (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.

Table 299. MCRE field descriptions(Continued)

Field	Description
16–18 n64Km	Number of 64 KB blocks in mid space. (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. (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. (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. (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.
27–28 n32Kl	Number of 32 KB blocks in low space. (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. (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.

29.3.2.3 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’.

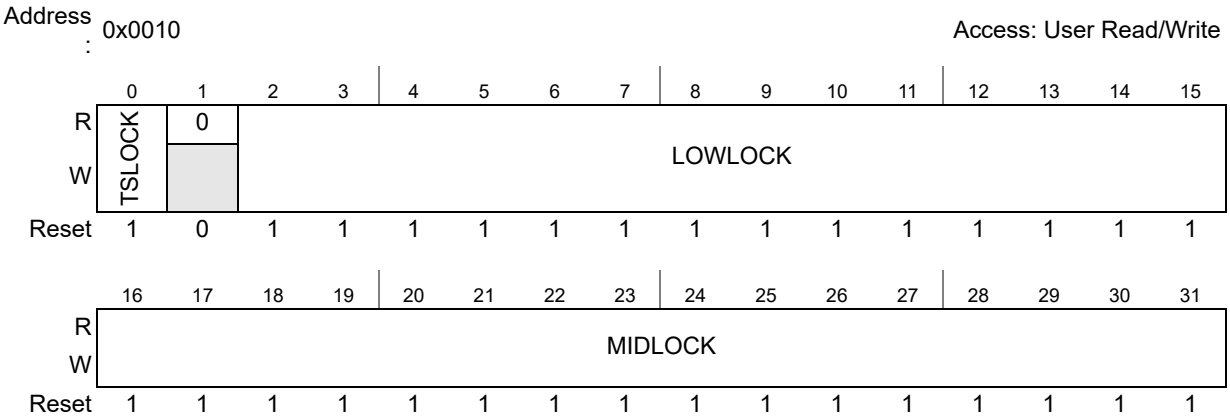


Figure 256. Locking 0 register (LOCK0)

Table 300. LOCK0 field descriptions

Field	Description
0 TSLOCK	UTEST NVM block lock (Read/Write) 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.
1	Reserved (read only). Writing to this bit has no effect and Reads always return 0.

Table 300. LOCK0 field descriptions(Continued)

Field	Description
2-15 LOWLOCK[13:0]	Low address space block lock 13-0 (Read/Write) 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[15:0]	Mid address space block lock 15-0 (Read/Write) 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.

29.3.2.4 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 are read as ‘1’.

To determine the effective status of the High Address Space, combine (Or) LOCK1 with sidebands

- f90_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 257. Locking 1 register (LOCK1)

Table 301. LOCK1 field descriptions

Field	Description
0–15	Reserved (read only) Writing to this bit has no effect and Reads always return 0.
16–31 HIGHLOCK[15:0]	High address space block lock 15-0 (Read/Write) 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.

29.3.2.5 Lock 2 register (LOCK2)

The LOCK2 (256K Address Space Block Locking) register provides a means of protecting blocks from being modified. The A256KLOCK bits corresponding to nonexistent blocks are read as ‘1’.

To determine the effective status of the 256KLOCK Address Space, combine (Or) LOCK2 with sidebands:

- f90_plock2 – in order to evaluate protection against program
- f90_elock2 – in order to evaluate protection against erase.

The effect of programming/erasing a block protected with LOCK2 is that no modification occurs and PEG is set to ‘1’.

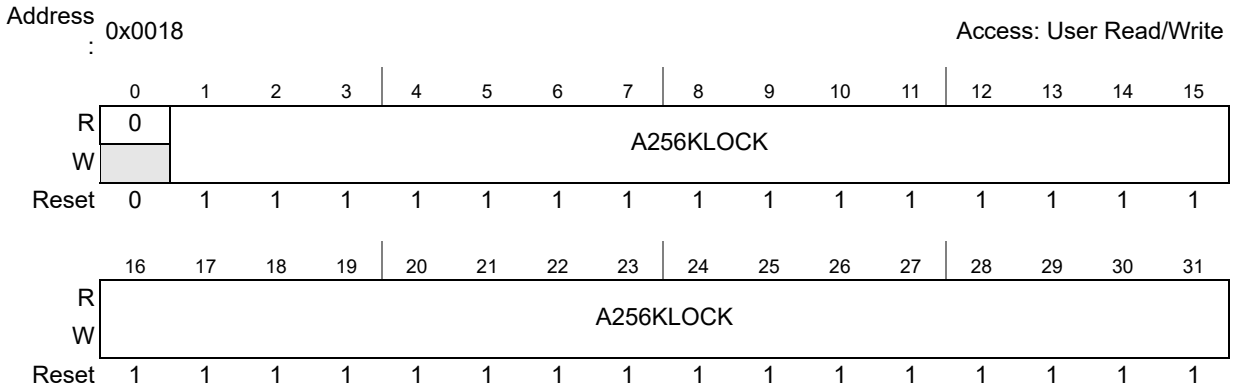


Figure 258. Locking 2 register (LOCK2)

Table 302. LOCK2 field descriptions

Field	Description
0	Reserved (read only). Writing to this bit has no effect and Reads always return 0.
1–31 A256KLOCK[30:0]	256K address space block lock 30-0 (Read/Write) These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the 256 KB blocks starts with A256KLOCK[0] and continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR[DONE] is set. – If a high voltage operation is suspended. The default value of the A256KLOCK bits is locked. When blocks are not present (due to configuration or total memory size), the A256KLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Low Address Space Block is unlocked and can be modified. 1 Low Address Space Block is locked and cannot be modified.

29.3.2.6 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 29.4.5.5, 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 259. Select 0 register (SEL0)

Table 303. SEL0 field descriptions

Field	Description
0-1	Reserved (read only). Writing to this bit has no effect and Reads always return 0.
2-15 LOWSEL[13:0]	Low address space block select 13-0 (Read/Write) 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[15:0]	Mid address space block select 15-0 (Read/Write) 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.

29.3.2.7 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 29.4.5.5, 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 260. Select 1 register (SEL1)

Table 304. SEL1 field descriptions

Field	Description
0–15	Reserved (read only). Writing to this bit has no effect and Reads always return 0.
16–31 HIGHSEL[15:0]	High/Data address space block select 1 15-0 (Read/Write) The 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.

29.3.2.8 Select 2 register (SEL2)

The SEL2 (256K Address Space Block Select 2) register provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL2 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR[ERS] (see step 4 in *Section 29.4.5.5, 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.

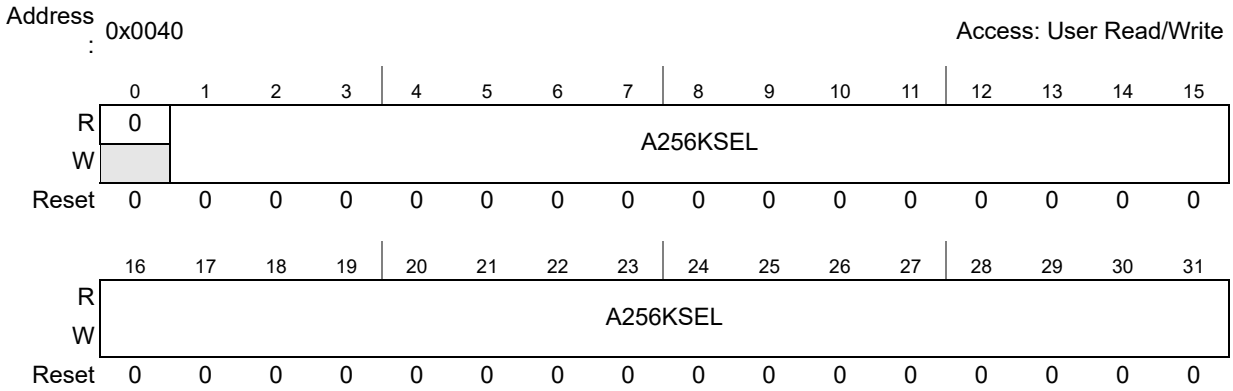


Figure 261. Select 2 register (SEL2)

Table 305. SEL2 field descriptions

Field	Description
0	Reserved (read only). Writing to this bit has no effect and Reads always return 0.
16–31 A256KSEL[30:0]	256K address space block select 30-0 (Read/Write) The block numbering for the 256 KB blocks starts with A256KSEL[0] and continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set – a high voltage operation is suspended. The select field must not be written when UT0[AIE] is high. When blocks are not present (due to configuration or total memory size) the corresponding A256KSEL bits default to unselected and are not writable. The reset value is always ‘0’, and register writes have no effect. 0 Low Address Space Block is unselected for Erase. 1 Low Address Space Block is selected for Erase.

29.3.2.9 Address register (ADR)

The Address register provides the first failing address in the event 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 306* according to the priority rules in *Table 307*.

Table 306. ADR update mode list

Mode	Update on error	Notes
Read	ECC Double Error, ECC Single Error (if UT0.SBCE=1)	—
Modify	FPEC error	—
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 307. 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 262. Address register (ADR)

Table 308. ADR field descriptions

Field	Description
0 SAD	Test Address (read only) When this bit is high, the address indicated by ADDR[23-3] belongs to the Test Block.
1–7	Reserved (read only). Writing to this bit has no effect and a read always returns 0.

Table 308. ADR field descriptions(Continued)

Field	Description
8–28 ADDR[23:3]	Address 23-3 (read only) 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.
29–31	Reserved (read only). Writing to this bit has no effect and a read always returns 0.

Table 309. 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 0x0003_FFFF	0x0100_0000 to 0x0103_FFFF
B0F2	0x0004_0000 to 0x0007_FFFF	0x0104_0000 to 0x0107_FFFF
B0F3	0x0008_0000 to 0x000B_FFFF	0x0108_0000 to 0x010B_FFFF
B0F4	0x000C_0000 to 0x000F_FFFF	0x010C_0000 to 0x010F_FFFF
B0F5	0x0010_0000 to 0x0013_FFFF	0x0110_0000 to 0x0113_FFFF
B0F6	0x0014_0000 to 0x0017_FFFF	0x0114_0000 to 0x0117_FFFF
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
B0UT	0x0080_4000 to 0x0080_5FFF	0x0040_0000 to 0x0040_1FFF
B0TF	0x0080_8000 to 0x0080_BFFF	0x0040_8000 to 0x0040_BFFF

Table 310. Flash Memory Partition 1 addresses mapping

Block	Flash format addresses	Absolute addresses
B1F0	0x0018_0000 to 0x0018_3FFF	0x0080_0000 to 0x0080_3FFF
B1F1	0x0018_4000 to 0x0018_7FFF	0x0080_4000 to 0x0080_7FFF
—	—	—
—	—	—
—	—	—

29.3.2.10 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	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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 263. User Test 0 register (UT0)

Table 311. UT0 field descriptions

Field	Description
0 UTE	User Test Enable (Read/Clear) 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 (Read/Clear) 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 also is used as an Enable for interrupt signals created by the embedded flash memory. ECC corrections that occur when SBCE is cleared are not logged.

Table 311. UT0 field descriptions(Continued)

Field	Description
2-21	Reserved (Read/Write). Writes have no effect. For compatibility purposes, any writes should be ‘0’.
22 NAIBP	Next Array Integrity Break Point (Read/Clear) 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. (Read/Write) 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.
24	Reserved
25 AISUS	Array Integrity Suspend (Read/Write). 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 (Read/Write) 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 (Read/Write) 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).
28	Reserved (Read/Write). Write this bit has no effect. It is suggested for compatibility to Write at 0.

Table 311. UT0 field descriptions(Continued)

Field	Description
29 AIS	Array Integrity Sequence (Read/Write) 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.
30 AIE	Array Integrity Enable (Read/Write) 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 (read only) 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 cannot be written, and is status only. 0Array Integrity Check is on-going. 1Array Integrity Check is done.

29.3.2.11 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.

29.3.2.11.1 User Multiple Input Signature 0 register (UM0)

UM0 represents bits 31–0 of the whole 144-bit word (2 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	MISR															
Reset	0	0	0	0	0	0	0	0	0	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	MISR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 264. User Multiple Input Signature 0 register (UM0)

Table 312. UM0 field descriptions

Field	Description
0–31 MISR[31:0]	Multiple input signature 31–0 (Read/Write). 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.

29.3.2.11.2 User Multiple Input Signature 1 register (UM1)

UM1 represents bits 63–32 of the whole 144-bit word (2 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	MISR															
Reset	0	0	0	0	0	0	0	0	0	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	MISR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 265. User Multiple Input Signature 1 register (UM1)

Table 313. UM1 field descriptions

Field	Description
0–31 MISR[63:32]	Multiple input signature 63–32 (Read/Write). 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.

29.3.2.11.3 User Multiple Input Signature 2 register (UM2)

UM2 represents bits 95–64 of the whole 144-bit word (2 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	MISR															
Reset	0	0	0	0	0	0	0	0	0	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	MISR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 266. User Multiple Input Signature 2 register (UM2)

Table 314. UM2 field descriptions

Field	Description
0–31 MISR[64:95]	Multiple input signature 95–64 (Read/Write). 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.

29.3.2.11.4 User Multiple Input Signature 3 register (UM3)

UM3 represents bits 127–96 of the whole 144-bit word (2 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	MISR															
Reset	0	0	0	0	0	0	0	0	0	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	MISR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 267. User Multiple Input Signature 3 register (UM3)

Table 315. UM3 field descriptions

Field	Description
0–31 MISR[96:127]	Multiple input signature 127–96 (Read/Write). 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.

29.3.2.11.5 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 2 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	MISR															
Reset	0	0	0	0	0	0	0	0	0	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	MISR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 268. User Multiple Input Signature 8 register (UM8)

Table 316. UM8 field descriptions

Field	Description
0–31 MISR[256:287]	Multiple input signature 287-256 (Read/Write) 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.

29.3.2.11.6 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	MISR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 269. User Multiple Input Signature 9 register (UM9)

Table 317. UM9 field descriptions

Field	Description
0–31 MISR[288]	Multiple input signature 288 (Read/Write) 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.

29.3.2.12 Over-program protection registers

Over-program protection prevents blocks from being over-programmed. These registers show the over-program protection status. See *Section 29.4.5.3, Over-programming protection (OPP) enable*.

29.3.2.12.1 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)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 270. Over-program protection 0 register (OPP0)

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications.

Table 318. OPP0 field descriptions

Field	Description
0–1	Reserved (read only) Writing to this bit has no effect and a read always returns 0.

Table 318. OPP0 field descriptions(Continued)

Field	Description
2–15 LOWOPP[13:0]	Low address space Over Program Protection 13–0 (Read) 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[15:0]	Mid address space Over Program Protection 15–0 (Read) 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.

29.3.2.12.2 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)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 271. Over-program protection 1 register (OPP1)

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications.

Table 319. OPP1 field descriptions

Field	Description
0–15	Reserved (read only) Writing to this bit has no effect and a read always returns 0.
16–31 HIGHOPP[15:0]	High address space Over Program Protection 15–0 (Read) 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.

29.3.2.12.3 Over-program protection 2 register (OPP2)

The 256K high Address Space is covered by the Over Program Protection 2 (OPP2) register.

Address: 0x0088 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	A256KOPP														
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 272. Over-program protection 2 register (OPP2)

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications.

Table 320. OPP2 field descriptions

Field	Description
0	Reserved (read only) Writing to this bit has no effect and a read always returns 0.
1–31 A256KOPP[30:0]	256K High address space Over Program Protection 1–31 (Read) The block numbering for the 256 KB Blocks starts at A256KOPP[0] and continues until all blocks are accounted for. The A256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the A256KOPP bits default to not over-programming protected and will remain 0. 0 High Address Space Block is unprotected from over-programming. 1 High Address Space Block is protected from over-programming.

29.3.2.13 Test Mode Disable Password Check register (TMD)

The Test Mode Disable Password Check (TMD) register provides a means of entering the password to be matched with the one stored in UTEST location in order to allow manufacturer access to protected Test Mode blocks by writing 0x5A4B_3C2D in the Test Mode Disable Seal location.

Address: 0x0090 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	PWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 273. Test Mode Disable Password Check register (TMD)

Table 321. TMD 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. See *Section 29.4.6.2, Test mode disable* for more information. Writes to the register have no effect except for a password challenge. Reads to this register will always return 0. The register's reset value is 0. **Note:** PWD bitfield is byte-reversed compared to UTEST TMD password : for instance, if UTEST TMD password is written 0x01020304, the PWD bitfield must be written 0x04030201

29.4 Functional description

This chapter describes the details of all the modes of operation available to the Flash Memory module.

29.4.1 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.

29.4.2 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]. 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] 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 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.

29.4.3 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 ‘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.

29.4.4 Read mode

Read mode is the active mode of operation of the Flash Memory module in which the Flash array can be read or ‘written’ (the so-called Interlock Write), or the Flash registers can be read or written.

The Flash Memory module enters Read mode when exiting from reset. The default state of the Flash Memory module is Read mode.

The main spaces (Low, Mid, High, and 256K) and Test address space of Flash array can only be read in Read mode.

Flash registers can even be read when the module is in Power-down or Low-power mode (except a few documented registers) as well as in Read mode. However, Flash registers can only be written while in Read mode.

Flash memory module is in Read mode, granting the same access time, under the following conditions:

- The Read state is active when the module is enabled (User Mode Read)
- The Read state is active when MCR[ERS] and MCR[ESUS] are high and MCR[PGM] is low (Erase Suspend)
- The Read state is active when MCR[PGM] and MCR[PSUS] are high (Program Suspend)
- The Read state is active when MCR[PGM] or MCR[ERS] are high and High Voltage operation is ongoing on one partition, thus MCR[EHV] is high (condition referred as Read-While-Write). In this case reads done on the other partitions are allowed, while reads done to the partition being operated on (either Erase or Program) result in an error with MCR[RWE] set and the data read back possibly being random including the possibility of ECC errors.

Code flash partitions reads return 128 bits (1 half-page = 2 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.

29.4.4.1 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.

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.

29.4.4.1.1 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 322 shows a set of doublewords sharing the same ECC value:

Table 322. 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 3 blocks for this purpose.

29.4.5 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 323*.

Table 323. 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 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.

29.4.5.1 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.

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 MCR[EHV] to start the internal program sequence or terminate via step 9.
5. Wait until 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 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.

Example 4. 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 */
```

29.4.5.2 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.

Example 5. 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.

Example 6. 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}.

29.4.5.3 Over-programming protection (OPP) enable

Over Programming Protection can be enabled on a block-by-block basis. In an OPP enabled block, any doubleword which has already been programmed cannot be programmed again. The OPP enable does not affect the Erase operation. It is possible to independently set OPP and (hardware) erase protection in order to avoid any kind of

overwrite. Using these protection mechanisms in conjunction with the Test mode disable feature gives even greater overwrite protection. Attempts to over-program result in MCR[PEG] being cleared.

OPP can be enabled for 16 KB, 32 KB, 64 KB and 256 KB blocks. The 256 KB blocks have selects available for the first four 256 KB blocks, and then a single select for the remainder of the 256 KB blocks.

OPP is enabled by sideband signals and, if enabled, the program operation is modified to check for programmed bits prior to undertaking a high voltage program event.

29.4.5.4 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.

29.4.5.5 Erase

Erase changes the value stored in all bits of the selected block(s) to logic ‘1’. An Erase sequence operates on any combination of blocks in the low, mid, high or 256K address space. The Test block cannot be erased.

The Erase sequence is fully automated within the Flash. The user only needs to select the blocks to be erased and initiate the Erase sequence.

Register locked blocks cannot be erased. Hardware locking which only affects the Erase operation is also available. Hardware locking is enabled by sideband signals and, if enabled, erases are prohibited on hardware locked blocks in the same way as erase software locking. If multiple blocks are selected for erase during an Erase sequence, no specific operation order can be assumed.

The Erase operation consists of the following sequence of events:

1. Change the value of MCR[ERS] 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 the MCR[EHV].
8. If more blocks are to be erased, return to step 2.
9. Write a logic ‘0’ to MCR[ERS] 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’.

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.

Example 7. 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 */
```

29.4.5.5.1 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 to1 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.

Example 8. 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.

Example 9. 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}.

29.4.5.5.2 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).

29.4.5.6 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).

29.4.5.6.1 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 UM0–9), to determine if an incorrect Read, or ECC detection was noted. Array integrity is controlled by the pfb clock (fclk) and 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 2 different Read accesses at the same location.

The whole check is done through 2 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 reg, but this does not change the final signature in the UM0–9.

Only data from existing locations are captured by the MISR. The MISR can be seeded to any value by writing the UM0–9 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 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.
9. Compare UM0–UM9 content with the expected result.
10. Write a logic '0' to UT0[AIE].

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.

Example 10. 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 256KSL2-1 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 */
```

29.4.5.6.2 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 2 different Read accesses at the same location. The whole check is accomplished through 2 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 reg, but this does not alter the final signature in the UM0–9.

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 UM0–9 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 the UT0[AID] bit 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 UM0–9 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.

Example 11. 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 */
```

29.4.6 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.

29.4.6.1 Modify Protection

The Flash Modify Protection information is stored in nonvolatile flash cells located in the UTEST region. This information is read once during the Flash Initialization phase following exit from reset and is stored in volatile registers that act as actuators at SoC level. The information is passed to the Flash Memory module through erase_lock and program_lock sidebands, and allows independent protection of each block of Low, Mid, High and 256K Address Space Block.

There are also Address Space Block Locking registers provided to independently lock/unlock each block of Low, Mid, High and 256K Address Space Block against program and erase. Locking is done through the LOCK0, LOCK1, and LOCK2 registers. These registers do not have a nonvolatile image stored in the UTEST area of TestFlash, so the locking information is kept on reset: volatile content is all 1's, meaning all blocks are locked.

All the volatile registers can be written to '0' or '1' at any time, therefore the user application can lock and unlock blocks when desired. Note that overall protection is achieved by the combination of sidebands with Low, Mid, High and 256K Address Space Block Locking registers in such a way that a block can be programmed/erased only if both (sidebands and Locking registers) indicate an unlocked state. Sidebands independently control program protection, erase protection and over-programming protection (OPP) on a block-by-block basis.

As UTEST and BAF (each one with its own lock bit, LOCK0[TSLOCK] and LOCK0[LOWLOCK] respectively) are two sections of the same physical block (TestFlash), effective locking is achieved via the 'AND' operand of the lock bits. For effective locking,

overall protection is obtained in conjunction with sidebands: UTEST and BAF have dedicated program protection sidebands (SoC specific).

29.4.6.2 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 0x0x90.

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 324. Test Mode Disable block select

Block	Block selection bits
Blocks in Mid space	Data[15:0]
Blocks in Low space	Data[29:16]
Blocks in High space	Data[47:32]
Blocks in 256K space	Data[78:48]

30 Decorated Storage Memory Controller (DSMC)

30.1 Introduction

This section details hardware support for atomic read-modify-write memory operations. In the Power Architecture, these capabilities are called “*decorated storage*”. It is supported by capabilities in the processor cores plus instantiations of a Decorated Storage Memory Controller (DSMC).

The Decorated Storage APU defines instructions for providing load and store operations to memory addresses that require *additional semantics* beyond just the reading and writing of data values to the addressed memory locations. Decorated storage operations are intended to be used for specific devices or memory targets that require these additional semantics. A “decoration” is the additional semantic information to be applied to the decorated storage operation by the DSMC.

Consider the basic mnemonics, syntax and formats for the decorated load instructions.

For loads, the syntax is `l[b,h,w]d{cb}x rT,rB,rA` where the data size specifier is defined as 8-bit (`b` = byte), 16-bit (`h` = halfword) or 32-bit (`w` = word), and the three register specifiers include `rT` as the destination target register, `rB` as the effective address and `rA` as the decoration. The optional `{cb}` specifier defines a version of the instruction which is treated as a cache bypass operation.

The syntax for store instruction is `st[b,h,w]d{cb}x rS,rB,rA` where the same data size specifiers are used, and the three register specifiers are `rS` as the source data register, `rB` as the effective address and `rA` as the decoration. The cache bypass attribute is again specified with the use of the optional `{cb}` in the instruction mnemonic.

For all decorated loads and stores, the memory effective address is simply defined by the rB register (hence the `“x”` mnemonic suffix, signaling an “indexed” addressing mode) and the decoration is specified by the rA register. The core transmits the contents of the rA register as the decoration value along with the access address (register rB) to the DSMC.

The decorated load and store instruction formats are shown in *Figure 274*.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
lbdx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	0	0	0	0	1	1	/
lhdx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	1	0	0	0	1	1	/
lwdx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	1	0	0	0	0	1	1	/
lbdcbx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	0	0	0	0	1	0	/
lhdcbx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	1	0	0	0	1	0	/
lwdcbx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	1	0	0	0	0	1	0	/
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
stbdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	0	0	0	0	1	1	/
sthdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	1	0	0	0	1	1	/
stwdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	1	0	0	0	0	1	1	/
stbdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	0	0	0	0	1	0	/
sthdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	1	0	0	0	1	0	/
stwdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	1	0	0	0	0	1	0	/

Figure 274. Power Architecture Decorated Load and Store Instruction Formats

The basic decorated load or store memory reference supplements the access address and attribute information with a 32-bit “decoration”. The DSMC decoration defines the special memory operation to be performed and optionally includes bit specifiers and/or operand data for the command.

The default format for the 32-bit decoration field defines a 4-bit command in the most significant bits (decoration[0:3]).

The cache bypass decoration load (`l[b,h,w]dcbx`) and store (`st[b,h,w]dcbx`) instructions do not distinguish between cacheable and cache-inhibited storage attributes, and instead have semantics to effectively emulate a cache-inhibited operation to storage regardless of the actual storage attributes. This is desired in order to allow a small portion of an otherwise cacheable storage area to be treated as non-cacheable using instruction supplied semantics, regardless of storage attributes provided by a memory management, protection or other memory control scheme. This allows decoration or other accesses to be applied to a portion of memory that is normally treated as cacheable according to its storage attributes, as well as to ensure that any stale copies of storage locations present in the cache are not used.

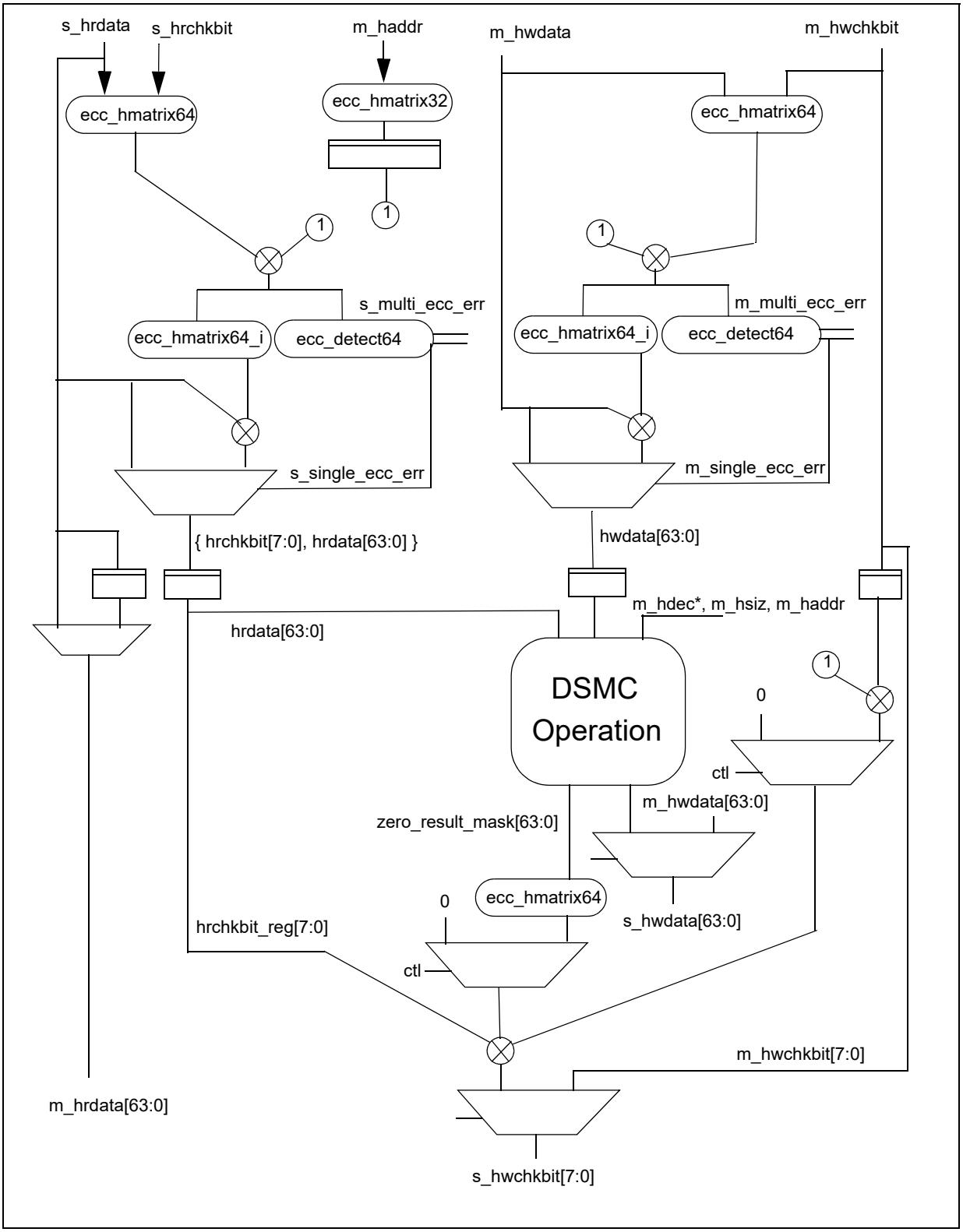
The combination of the decorated load and store instruction support in the e200zX cores plus the decorated storage memory controller in the core platform adds a robust atomic read-modify-write capability to the Power Architecture. The architecture capability defined by these core and platform functions is targeted at manipulation of *n*-bit fields in peripheral registers (and is consistent with I/O hardware addressing in the Embedded C standard) as well as software synchronization data structures needed in multi-core systems (mutexes, semaphores, test-and-set and compare-and-swap structures).

For additional information on the decorated load and store instructions from the core's perspective, refer to the appropriate e200zX core reference manual.

30.2 DSMC Block Diagram

The following block diagram, *Figure 275*, shows the basic datapaths for the ECC logic in the DSMC. On both the read data path (hrdata) and the write data path (hwdata) there is logic to detect an ECC error and correct the data. The circle with X inside represents an XOR operation. The ecc_hmatrix and ecc_detect blocks are coefficient matrixes based on the Hsiao ECC algorithm. Notice in the recalculation of the s_hwchkbit[7:0] that there is a read chkbit from the slave (hrchkbit), a zero padded result contribution, and a contribution from the hwchkbit from the master. The details of the DSMC bit field and other operations are not shown (represented by the DSMC Operation box).

Figure 275. DSMC Block Diagram



30.3 Decorated Stores: `st[b,h,w]d{cb}x rS,rB,rA`

The next sections present descriptions of the specific operations, based on the 4-bit command field defined in decoration[0:3]. These descriptions include the bit pattern definitions for the 32-bit decoration value and include pseudo-code detailing the sequence of operations. The decoration formats are defined using a `<command>.<size>` syntax where the operand size specifier is b (byte, 8-bit), h (halfword, 16-bit) or w (word, 32-bit). The operand size is specified directly in the decorated instruction executed in the core, and this attribute is driven to the system bus as part of the decorated data transfer. Additionally, the write data is taken from the right-justified 8, 16 or 32 bits of the rS register, that is:

```
 8-bit wdata = rS[24:31]
16-bit wdata = rS[16:31]
32-bit wdata = rS[ 0:31]
```

Likewise, read data is loaded into the right-justified 8, 16 or 32 bits of the rT register:

```
 8-bit rdata = rT[24:31]
16-bit rdata = rT[16:31]
32-bit rdata = rT[ 0:31]
```

For the byte and halfword decorated load instructions, the upper bits of the rT register are zero filled.

The starting bit (SRTBIT) position follows the Power Architecture convention where the MSB is bit 0 and the LSB is bit 31. The bit field width (BFW) value defines the width and BFW = 0 specifies the maximum width, that is, the container size.

The basic decorated store includes three data transfer fields sourced from the core: the access address (rB), the decoration (rA) and the write data (wdata) operand (rS). There are five operations defined and most of these transactions convert a single core AHB write bus cycle into an atomic read-modify-write, that is, an indivisible read followed by write sequence. Support for three basic boolean logic functions (AND, OR, XOR) is provided along with a compare-and-store operation plus a bit field insert operation. These operations do not support any type of bit field wrapping.

In the next sections detailing the decorated store operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- wdata = rS

30.3.1 Bit Field Insert (BFINS) into an 8-, 16- or 32-bit Memory Container

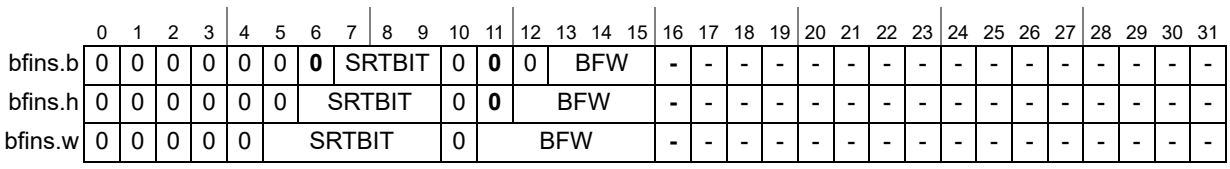


Figure 276. Decoration Format: BFINS

This command inserts a bit field defined by SRTBIT and BFW into the memory “container” defined by the access size associated with the decorated store instruction using an atomic read-modify-write sequence.

```
tmp = mem[accessAddress, size] // memory read

if bfw == 0
   then bfw = container

if ((srtbit + bfw) <= container)// generate bit mask
    mask = ((1 << bfw) - 1) << (container - (srtbit + bfw))
else
    mask = ((1 << bfw) - 1) >> ((srtbit + bfw) - container)

tmp = tmp   & ~mask// modify
    | wdata &  mask

mem[accessAddress, size] = tmp// memory write
```

The write data operand (wdata) associated with the decorated store instruction contains the bit field to be inserted. *It must be properly aligned within a right-justified container in the source register (rS)*, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword or the entire 32 bits for a word operation.

To illustrate, consider the following example of the insertion of the 3-bit field “xyz” into an 8-bit memory container, initially set to “abcd_efgh”. For all cases, the bit field width (BFW) is 3.

```
if SRTBIT = 0 and the decorated store rS register[24:31] = xyz-_----,
   then destination is "xyzd_efgh"
if SRTBIT = 1 and the decorated store rS register[24:31] = -xyz_----,
   then destination is "axyz_efgh"
if SRTBIT = 2 and the decorated store rS register[24:31] = --xy_z---,
   then destination is "abxy_zfgh"
if SRTBIT = 3 and the decorated store rS register[24:31] = ---x_yz--,
   then destination is "abcx_yzgh"
if SRTBIT = 4 and the decorated store rS register[24:31] = ---_xyz-,
   then destination is "abcd_xyzh"
if SRTBIT = 5 and the decorated store rS register[24:31] = ---_-xyz,
   then destination is "abcd_exyz"
if SRTBIT = 6 and the decorated store rS register[24:31] = ---_--xy,
   then destination is "abcd_efxy"
if SRTBIT = 7 and the decorated store rS register[24:31] = ---_---x,
   then destination is "abcd_efgx"
```

If the bit field insert operation specifies SRTBIT as 0 and BFW as 0 (indicating the width matches the container width), then the operation is logically equivalent to a simple register store operation.

30.3.1.1 Compare-and-Store (CAST)

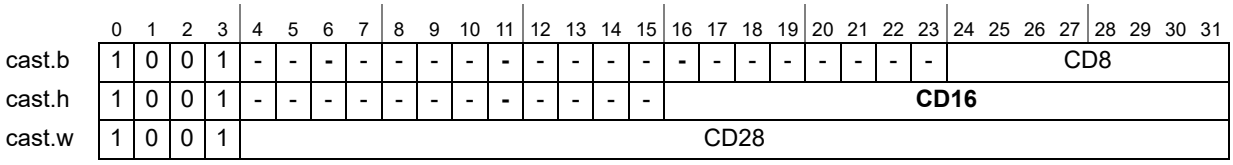


Figure 277. Decoration Format: CAST

This command begins by performing a read of the referenced memory address and comparing that data with the operand included in the decoration word. If the read data is equal to the compare operand, the write data associated with the decorated store instruction is placed into the memory location, else the original read data is rewritten into the memory location. The write operand is selected from the appropriate data byte lanes in the same manner as any memory store operation; this implies the write data operand is right justified in the rS source register.

```
tmp = mem[accessAddress, size] // memory read

if (size == 8)// define compare_operand
    compare_operand = CD8
else if (size == 16)
          compare_operand = CD16
     else compare_operand = {0x0, CD28}// zero-filled data

if (tmp == compare_operand)// compare - "modify"
     mem[accessAddress, size] = wdata// memory write
else mem[accessAddress, size] = tmp
```

Note that for the word size operation (cast.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32-bit compare operand.

30.3.1.1.1 Logical AND (AND)

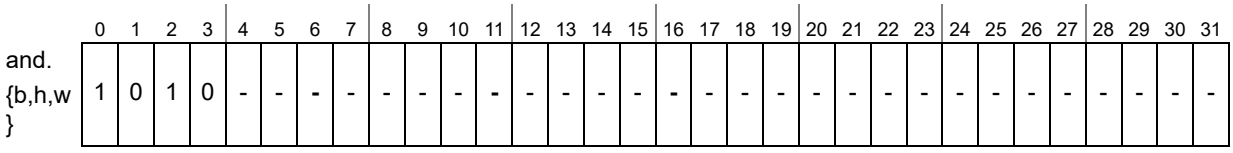


Figure 278. Decoration Format: AND

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical AND operation using the write operand defined by the rS register; finally, the result of the AND operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp & wdata// modify
mem[accessAddress, size] = tmp// memory write
```

30.3.1.1.2 Logical OR (OR)

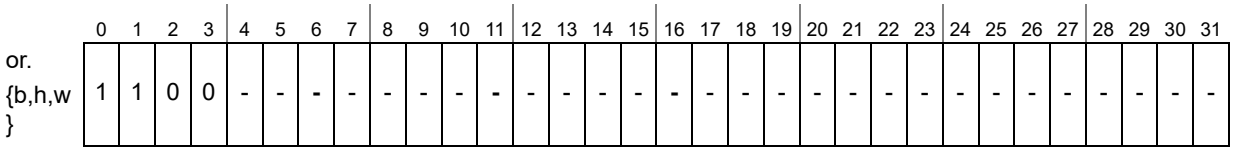


Figure 279. Decoration Format: OR

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical OR operation using the write operand defined by the rS register; finally, the result of the OR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp | wdata// modify
mem[accessAddress, size] = tmp// memory write
```

30.3.1.1.3 Logical Exclusive-OR (XOR)

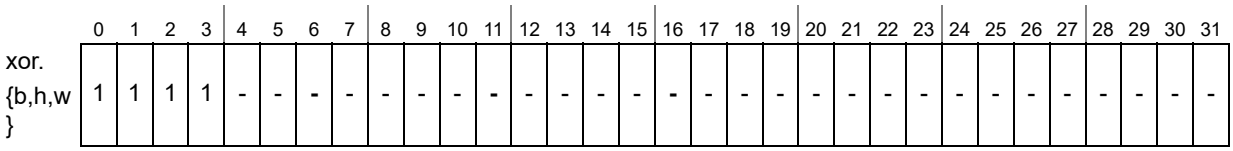


Figure 280. Decoration Format: XOR

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical XOR operation using the write operand defined by the rS register; finally, the result of the XOR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp ^ wdata// modify
mem[accessAddress, size] = tmp// memory write
```

30.3.1.2 Decorated Loads: `l[b,h,w]d{cb}x rT,rB,rA`

The basic decorated load includes two data transfer fields sourced from the core: the access address (rB), the decoration (rA) plus the read data (rdata) operand returned from DSMC to the core and loaded into the destination register (rT). There are 3 operations defined and two of these transactions convert a single core AHB read bus cycle into an atomic read-modify-write. Support for a swap function, a load-and-set-1-bit functions are provided along with a simple load (aka an “extract”) operation.

In the next sections detailing the decorated load operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- rdata = rT

30.3.1.2.1 Simple Load (SLD)

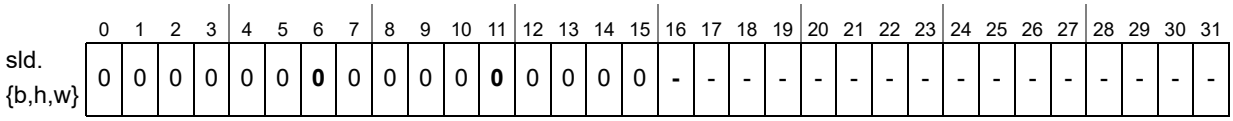


Figure 281. Decoration Format: SLD

This command performs a simple memory load of the reference size from the access address. For this operation, the 12-bit field defined in decoration[4:15] must be zeroes else the transfer is not performed and error terminated.

```
rdata = mem[accessAddress, size] // memory read
```

Support for generic bit field extract operations is best handled using existing core and compiler capabilities involving standard load instructions followed by left and right shift operations to emulate these functions.

30.3.1.2.2 Registers-and-Memory Exchange (SWAP)

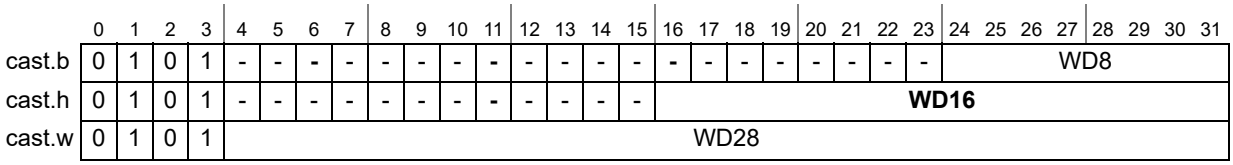


Figure 282. Decoration Format: SWAP

This command loads the contents of the referenced memory location into the core's rT register and stores the source operand from the core's rA register (the decoration) into the memory location.

```
tmp    = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp

if (size == 8)// store_operand - "modify"
    store_operand = WD8
else if (size == 16)
          store_operand = WD16
     else store_operand = {0x0, WD28}// zero-filled write data

mem[accessAddress, size] = store_operand// memory write
```

Note that for the word size operation (swap.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32-bit store operand.

30.3.1.2.3 Load-and-Set-1(Bit) (LAS1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
las1.b	0	1	1	0	0	0	**0**	BIT			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.h	0	1	1	0	0	0	BIT				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.w	0	1	1	0	0	BIT					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 283. Decoration Format: LAS1

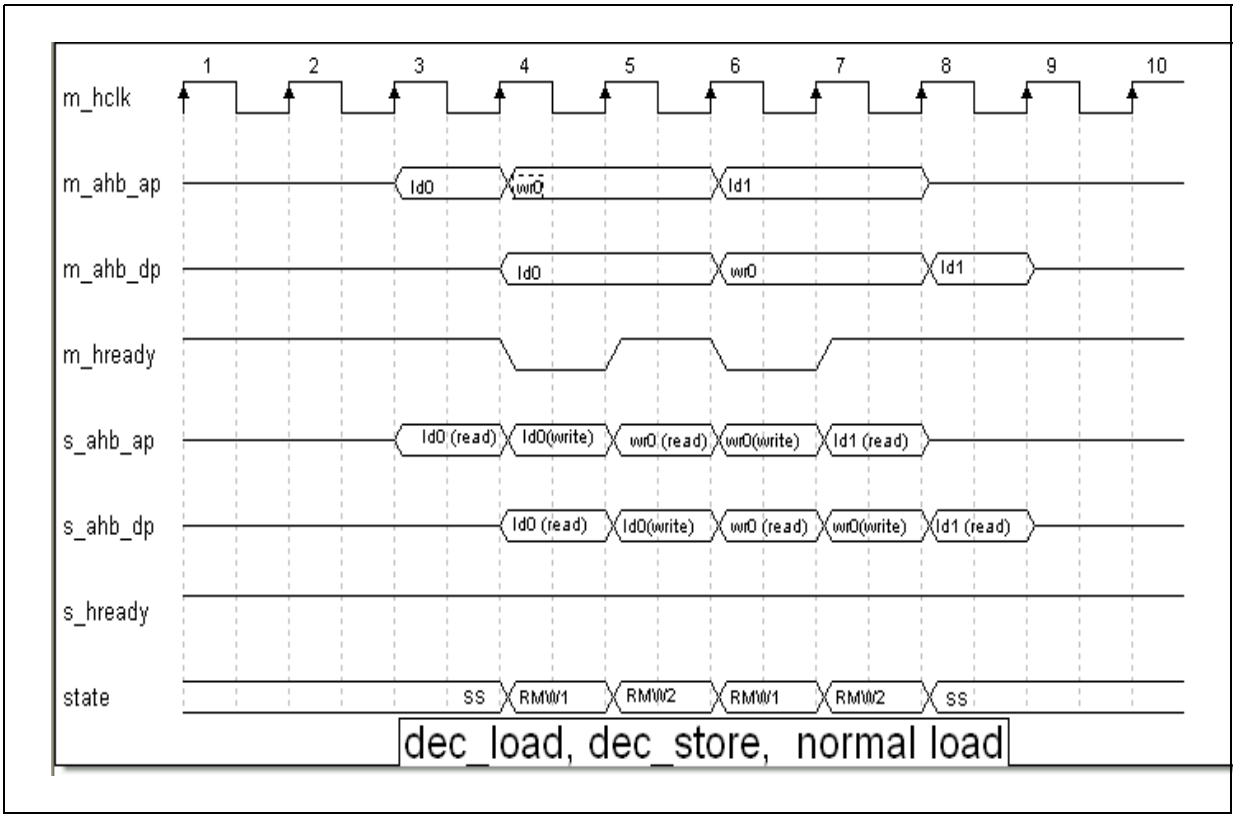
This command first reads the referenced memory location; it then modifies the read operand by setting the single bit position defined in the decoration (BIT); the modified data is then written back to the referenced memory location and the original read data returned to the initiating processor core.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp
mask  = 1 << (container - bit - 1)// generate bit mask
tmp   = tmp | mask// modify
mem[accessAddress, size] = tmp// memory write
```

30.4 DSMC Timing Diagram

The following timing diagram, *Figure 284*, shows a decorated load followed by a decorated store instruction which is then followed by a normal load instruction. Both decorated instructions are read-modify-write instructions and thus they take at least two cycles to execute. In the first cycle an ECC calculation is performed on the read data as well as the write data. Then in the second cycle of the decorated instruction the decorated operation is performed and if it is a decorated store, the result is merged with the read data and sent to the write data bus along with the recalculated ECC write chkbits.

Figure 284. DSMC Timing Diagram



30.5 DSMC Instantiations

The decorated storage memory controller which performs these operations is instantiated multiple times within the core platform and physically resides between the slave ports of the crossbar and the targeted memory controllers.

The module includes error checking logic that validates the decoration field is properly defined (all illegal commands are rejected and the transfer error terminated). Additionally, the decorated memory controller only operates on aligned, single data transfers (misalignment and/or bursts are not supported and error terminated if attempted).

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

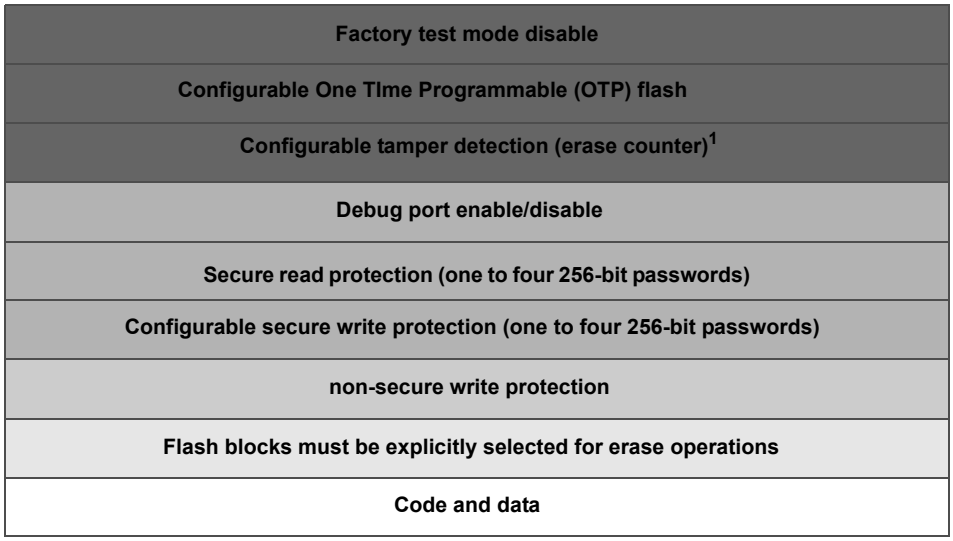
31 Flash Memory Programming and Configuration

This chapter provides basic guidance on activities related to configuring and programming the embedded flash memory in the SPC572Lx microcontroller.

Multiple layers of protection are available to prevent unintended updates to flash contents and unauthorized reads and writes to flash contents. The protection mechanisms range from simple write-protect bits for individual flash blocks to four levels of 256-bit password read and/or write protection and configuration of flash memory blocks as One Time Programmable (OTP).

Additionally, the SPC572Lx microcontroller includes a debug port enable/disable, and the ability to disable the ability for the device to be put into factory test mode.

Figure 285. Flash memory protection layers



Note:

1. Tamper detection module (TDM) is not supported on SPC572Lx.

At the lowest layer of protection, all flash erase operations require the explicit selection of one or more blocks of flash memory before the operation is performed. When executed, the operation is only performed on the selected block(s).

The next layer is the chip's ability to lock individual flash blocks against unintended programming and erase operations. This level of protection does not offer any password protection and is intended to provide protection against *accidental* changes to code and data.

The first layer of *secure* protection is the microcontroller's ability to support up to four levels of 256-bit password-secured *write* protection for individual flash blocks.

Similarly, up to four levels of password-secured *read* protection can be implemented. Unlike the password-secured write protection, which is implemented on a per-block basis, password-secured read protection applies to five pre-defined (at the factory) groups of flash

blocks. The read protection applied to a group applies to each block within the group. In contrast, secure write protection is defined at the individual block level.

Secure password protection can also be implemented for the debug port.

Another protection feature is the ability to configure any flash block as one-time programmable (OTP).

Finally, the microcontroller includes a feature that allows the customer to disable the ability to place the device in factory test mode.

The remainder of the chapter provides details regarding these protection mechanisms and gives a brief overview of security configuration planning.

Note: Most of the topics in this chapter build on information presented in previous topics. To get a clear overview of how the flash protection features work and relate to each other, read the chapter in sequence from beginning to end.

Access protection configuration is applied on a per-block basis (except read protection applied to groups of flash memory blocks), i.e., a custom level of access protection can be provided for each flash memory block. This is accomplished through the use of registers containing bits that are mapped to individual flash blocks. Understanding the mapping is key to understanding the implementation and use of the SPC572Lx microcontroller's flash protection features.

Depending on the type of protection to be applied, different sets of registers (or even DCF records) are used, but the mapping scheme is consistent. This mapping scheme is first presented in *Section 31.1: Selection of flash memory blocks for erase*. It is important to understand the topic before proceeding to other sections.

31.1 Selection of flash memory blocks for erase

The registers used for selecting flash blocks for erase operations are the flash control registers named LOCK0, LOCK1, and LOCK2. There is also a LOCK3 register, but no flash blocks are mapped to it.

The LOCK0–2 registers are 32-bit registers containing fields associated with specific address spaces: low, mid, high, and 256 KB. Each non-reserved bit in the fields is mapped to a specific flash block.

For example, *Figure 286* shows the Select 0 (SEL0) register for the flash module. This register provides a way of selecting blocks in the low and mid address spaces for erase operations.

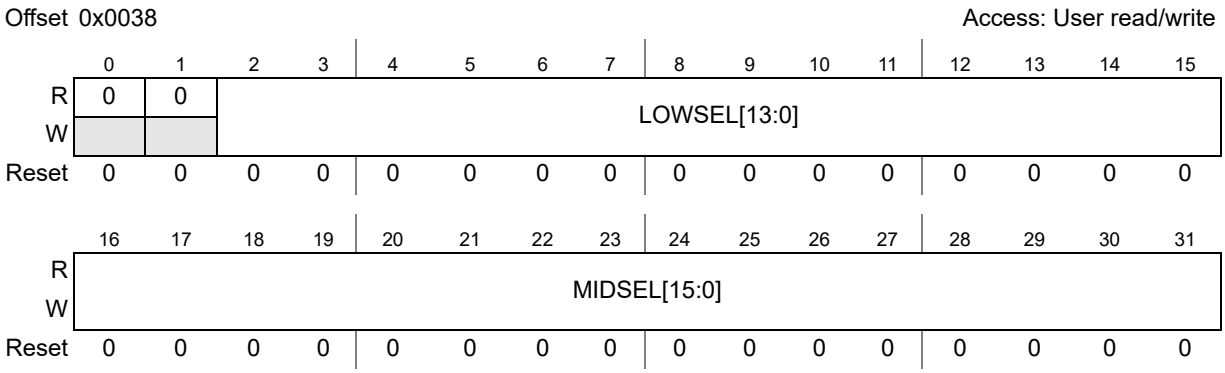


Figure 286. SEL0 flash control register

Table 325 shows an example of a mapping of SEL0 register's LOWSEL and MIDSEL field bits to individual flash blocks[g]. In this case, not all bits of the LOWSEL and MIDSEL fields are mapped—only LOWSEL bits 0–11 (register bits 15–4[h]) and MIDSEL bits 0–2 are mapped. Before any mapped block can be erased, it must be selected by writing a '1' to its corresponding bit in one of the SEL*n* registers. See the device configuration chapter for the complete mapping.

Note: Values written to the SEL registers do not persist through device reset. As indicated by the reset values shown in Figure 286, all bits are reset to '0'. As a result, after a device reset all blocks mapped to the register are, by default, not selected for erase.

Table 325. Sample low and mid address space flash block bit mapping

Start Address	End Address	Size (KB)	Partition	LOCK*n* (1) Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit(2)	Read Lock group (3)	Block No.	Description
0x0040_4000	0x0040_7FFF	16	0	LOCK0	LOWLOCK[0]	16	—	0	BAF(4)
0x00FC_0000	0x00FC_3FFF	16	0	LOCK0	LOWLOCK[1]	17	1	1	Code Flash(5)
0x00FC_4000	0x00FC_7FFF	16	0	LOCK0	LOWLOCK[2]	18	1	2	Code Flash(4)
0x00FC_8000	0x00FC_BFFF	16	1	LOCK0	LOWLOCK[3]	19	1	3	Code Flash(4)
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash(4)
0x00FD_0000	0x00FD_7FFF	32	0	LOCK0	LOWLOCK[6]	22	1	6	Code Flash
0x00FD_8000	0x00FD_FFFF	32	1	LOCK0	LOWLOCK[7]	23	1	7	Code Flash
0x00FE_0000	0x00FE_FFFF	64	0	LOCK0	LOWLOCK[8]	24	1	8	Code Flash
0x00FF_0000	0x00FF_FFFF	64	0	LOCK0	LOWLOCK[9]	25	1	9	Code Flash

g. The table lists the LOWLOCK register field but the structure is identical to the LOWSEL field in *Figure 286*.

h. Register bits in Power Architecture-based microcontrollers are numbered with the most significant bit as 0. This is not necessarily the case with register field bit numbering. The field bit numbering shown in the figure has the least significant bit numbered as 0.

Table 325. Sample low and mid address space flash block bit mapping(Continued)

Start Address	End Address	Size (KB)	Partition	LOCK*n* (1) Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit(2)	Read Lock group (3)	Block No.	Description
0x0040_0000	0x0040_3FFF	16	0	LOCK0	TSLOCK	31	0	32	UTEST(6)

1. There are multiple sets of LOCK*n* registers, i.e., the flash module's LOCK0–-3 registers and PASS module's PASS_LOCK0–3_PG0–3, but all use the same mapping of register bits to flash blocks. The same mapping applies the flash module's SEL0–3 registers.
2. The DCF record DATA bit refers to the DATA[31:0] field contained in DCF records created during OEM configuration activities. The DCF records related to flash blocks are used to implement password-secured write protection, password-secured read protection, tamper detection, and debug port enable/disable as a permanent part of the device configuration.
3. The Read Lock group pertains to protecting code and data blocks from being read by debuggers. This type of protection is implemented using DCF records and the PASS_LOCK3_PG0–3 registers.
4. The Boot Assist Flash (BAF) is one time programmable and the majority of it is programmed in the factory. The BAF handles the initial start up of the device.
5. This block is searched to determine the boot location (if it contains the boot header).
6. One time programmable.

Example 12. Selecting blocks for erase

Referring to *Table 325*, writing the value 0x0030_0000 to the SEL0 register (*Figure 286*) selects the block Start Address: 0x00FC_C000 to End Address: 0x00FC_FFFF for erase.

Note: The mappings of SELn register bits to flash blocks vary with different microcontrollers. See the flash memory section in the Device Configuration chapter of this reference manual for the mappings specific to the SPC572Lx microcontroller.

31.2 Non-secure write protection

Non-secure write protection refers to the chip's ability to lock individual flash blocks against programming and erase operations. This level of protection is intended to provide protection against *accidental* changes to code and data. Although similar in function, this is *not* associated with the built-in register protection (REG_PROT) feature included with many of the chip's modules. The non-secure write protection feature detailed here applies only to embedded flash memory.

Implementing non-secure write protection is almost identical to selecting specific flash blocks for an erase operation—the only difference is the register used. To provide non-secure write protection for a block, write a '1' to the corresponding mapped bit found in one of the LOCK0–2 flash control registers.

Note: As indicated by the reset values shown in Figure 287, after reset all blocks mapped to the register are, by default locked against programming and erase. Values written to the LOCK and SEL registers do not persist through device reset.

The same bit mapping that applies to the SEL0–3 registers also applies to the LOCK0–3 registers.

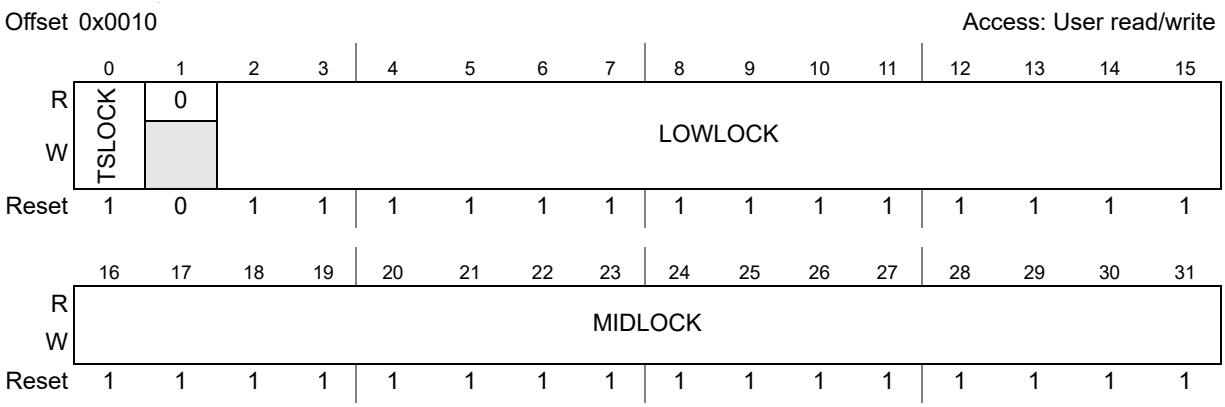


Figure 287. LOCK0 flash control register

Example 13. Selecting

Referring to *Table 325*, writing the value 0x0030_0000 (same value used in *Example 12*) to the LOCK0 register (*Figure 287*) locks the block Start Address: 0x00FC_C000 to End Address: 0x00FC_FFFF (same block selected in *Example 12*) so they cannot be programmed or erased, even if selected for erase.

As long as the LOCK0[LOWLOCK] bits are unchanged, the above blocks cannot be modified unless the protection is overridden (discussed in *Section 31.3: Secure write protection*). All other blocks mapped to bits in the LOCK0 *can* be programmed because their corresponding bits have been set to '0'.

31.3 Secure write protection

Secure write protection refers to the chip's ability to implement up to four levels of 256-bit password-secured *write* protection for individual flash blocks. Use of this feature is optional but the default behavior of the SPC572Lx microcontroller is to apply secure write protection. If there is a need to *not* write-protect specific blocks, that needs to be configured by creating the appropriate DCF records.

Note that the *secure* write protection settings override the non-secure write protection detailed in *Section 31.2: Non-secure write protection*.

Implementing a custom secure write protection scheme requires creating DCF records that define the level of write protection to be applied to each flash block. The DCF records use a bit mapping scheme identical to the mapping in the flash module's SEL*n* and LOCK*n* registers. The write protection scheme becomes a permanent part of the device but blocks can be temporarily unlocked. Depending on the specific implementation, temporarily unlocking the write protection requires matching up to four 256-bit passwords.

The easiest way to understand the secure write protection functionality is to examine the role played by the PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers.

The PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers are four identical sets of registers contained in the PASS module—they distinguished by the PG*n* suffix. The PG*n* (Password Group *n*) suffix indicates the password required to unlock that set of registers. Password 0 unlocks PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. Similarly, the registers suffixed with PG1 are unlocked by matching password 1, and so on.

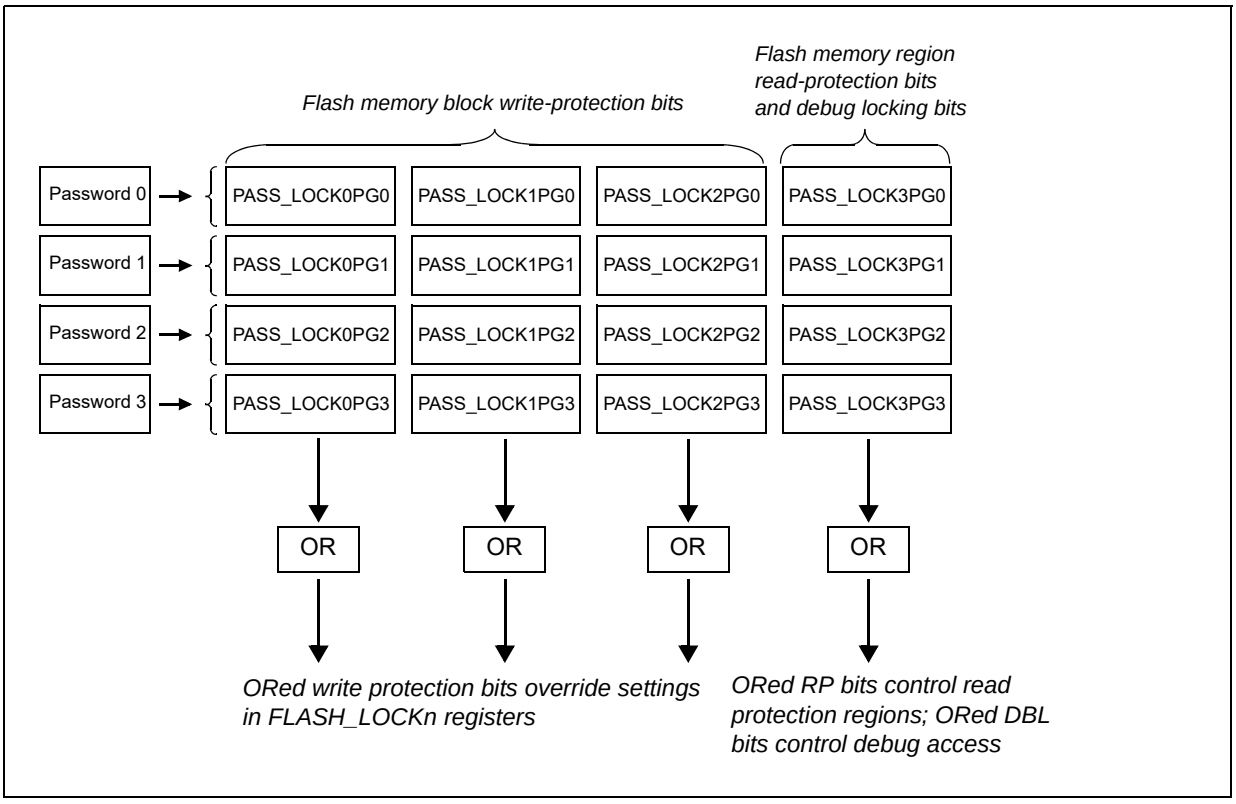
The PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* register sets have two functions related to secure write protection:

- Reading the registers provides the secure write lock status for each flash block.
- Changing the value of a mapped bit in a register changes the secure write protect status of the corresponding flash block.

Each set of registers contains the same mapping of bits to flash blocks as the flash module's SEL*n* and LOCK*n* registers. In any of the four PASS_LOCK*x*_PG*n* register sets, if a mapped bit is set to '1', locking a flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access to the register, and setting the bit to '0'. If the same bit is set to '1' in more than one PASS_LOCK*x*_PG*n* register set, each must be unlocked using the corresponding password and the bit must be set to '0' before the flash block is available for programming or erase.

Figure 288 shows how the registers interact. Note that secure write protection is only affected by the settings in PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n*. The PASS_LOCK3_PG*n* registers have additional security functions, discussed in later topics.

Figure 288. Secure write protection



The next sections discuss implementing secure write protection and overriding secure write protection.

31.3.1 Implementing secure write protection

Implementing secure write protection is a one-time task performed either at the factory or by the customer during chip configuration activities. It normally requires creating a DCF record for each of the PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n* registers(i). The DCF records

have a structure that includes the same mapping of bits to flash memory blocks as the flash module's SEL*n* and LOCK*n* registers. The values assigned become the permanent register reset values for the PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n* registers.

See *Chapter 8: Device Configuration Format (DCF) Records* for complete details on DCF records. This section summarizes the contents of a DCF record as required to implement secure write protection.

Even though the mapping of bits to flash memory blocks is identical to the mapping used for selecting flash blocks for erase of non-secure write locking, there are two significant differences:

1. Configuring flash blocks for secure write protection is done only during device configuration to customer specifications, either at the factory or by the customer during OEM production activities. The selection becomes a permanent part of the device configuration.
2. Configuration of blocks for secure write protection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

Caution: Careful planning is required before writing DCF records. They are created in a write-once area of flash. Multiple instances of any DCF record can be created, but in some cases only the record created first is used and with others only the most-recently written record is used. Be sure to understand the characteristics of a DCF record before creating it.

The DCF records required for implementing a secure write protection scheme are detailed in the PASS section of DCF client list table (*Table 71: DCF client list* in *Chapter 8: Device Configuration Format (DCF) Records*). Table entries provide the CS and Address value combinations that correspond to PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers.

The format of a secure write protection record is summarized here for convenience.

DCF records appear as contiguous double-word (64-bit) entries programmed in a reserved area of OTP UTEST flash memory. The structure of a DCF Record is shown in *Figure 289*.

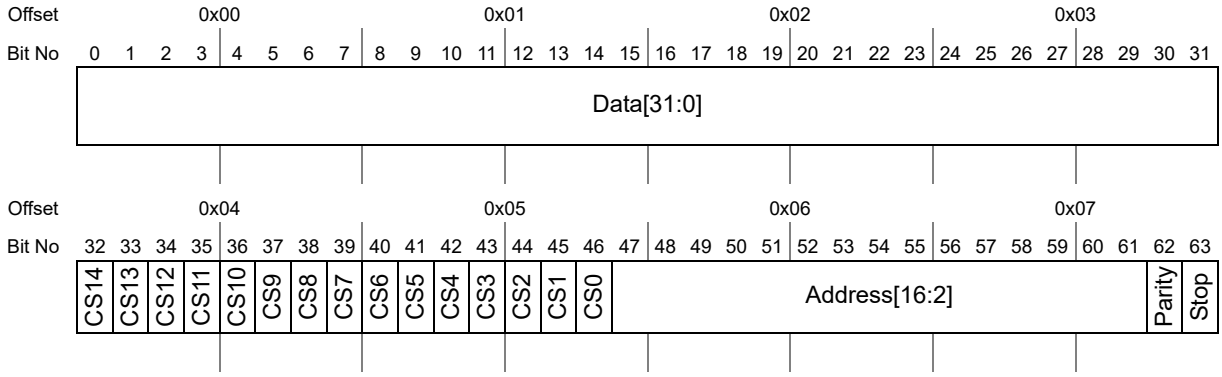


Figure 289. DCF record structure

i. The default behavior for secure write protection in the case where passwords are created but no secure write protection DCF records are created, is for all blocks to have four levels of password-secured write protection. To vary the protection among all blocks, a DCF record for each password group must be created.

Note: The values in the PASS_LOCKxPGn registers override the settings in the flash module's LOCK0–3 registers.

Values written to the PASS_LOCKxPGn registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated flash block to be locked against programming or erase.

The PASS_LOCKxPGn bits take effect when the life cycle is matured to "OEM Production" or older.

Example 14. Implementing secure write protection

Implementation of secure write protection is done during initial flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) must be stored in a protected OTP area of device flash called UTEST flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n*) that have the same mapping of register bits to flash blocks as the flash module's SEL0–3 registers and LOCK0–3 registers.
2. DCF records must be created to define which flash memory blocks are to be locked and which blocks are not locked.

The records defined in step 2, determine the reset values of the PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers and therefore the write protection settings that take effect each time the microcontroller is reset.

Assume four passwords have been previously implemented and the block Start Address: 0x00FC_C000 to End Address: 0x00FC_FFFF is to be locked in the password 0 group.

From the PASS section of *Table 71: DCF client list* in *Chapter 8: Device Configuration Format (DCF) Records*:

Table 326. DCF client list of PASS

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0000	LOCK0_PG0

Referring to *Figure 289*, the values for the DCF record are:

- Data[31:0] is 0x0030_0000 (bits 0–31) (same value used in previous examples)
- CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
- Address[16:2] is 0b000_0000_0100_0000 (bits 47–61)
- Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
- Preferred Stop value is 0 (see footnotes in *Figure 290*.

The following DCF record provides secure write protection for the blocks in *Table 325* by setting the reset values PASS_LOCK0_PG0 to the appropriate value to lock the blocks.

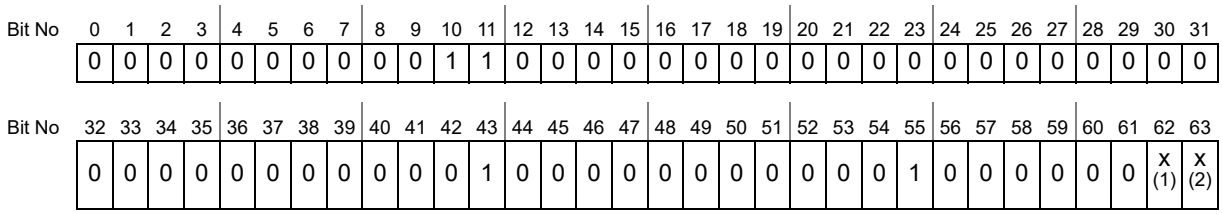


Figure 290. Sample DCF record for secure write protection

1. Bit 62 indicates parity strategy, which is not used in the PASS DCF records so it is a “don’t care”.
2. Bit 63 indicates whether the current record is the last DCF. Generally, this should be written as ‘0’ so the first unprogrammed (all 1’s) DCF record acts as the stop record.

The remaining blocks mapped to the PASS_LOCK0_PG*n* registers do *not* have secure write protection provided by the password 0 register set because the DATA field of DCF record has only the bits mapped to flash blocks 4 and 5 set to ‘1’.

Caution: The above example is intended to highlight the need to carefully understand how DCF records work. The protected blocks are secured by password 0 but by default they are also secured by passwords 1, 2, and 3. In contrast the blocks *not* secured by password 0 in this example *are* secured by passwords 1, 2, and 3.

See the PASS section of DCF client list table (*Table 71: DCF client list* in *Chapter 8: Device Configuration Format (DCF) Records*) for details on the DCF records.

31.3.2 Overriding secure write protection

The “PG*n*” (Password Group *n*) suffix on each the register names indicates the password each set of registers is associated with. For example, password 0, is used to unlock PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. The bits in the first three registers (PASS_LOCK0_PG0, PASS_LOCK1_PG0, and PASS_LOCK2_PG0) are mapped exactly the same as the bits in the flash module’s SEL0–2 registers and LOCK0–2 registers.

Like the SEL0 and LOCK0 registers, the PASS_LOCK0_PG*n* registers all contain bits mapped to the low and mid address spaces, and the mapping scheme is the same. A mapped bit set to ‘1’ in any of the registers causes the corresponding block to be locked against program or erase. A mapped bit set to ‘0’ in any of the registers makes the corresponding block available for programming or erase.

If a bit is set to ‘1’, locking a flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access, and setting the bit to ‘0’. If the same bit is set to ‘1’ in more than one of the PASS_LOCK*x*PG*n* register sets, each must be unlocked using the corresponding password and the bit must be set to ‘0’ before the flash block is available for programming or erase.

Figure 291 shows the PASS_LOCK0_PG*n* register. As you can see, it has the same structure as the flash module’s LOCK0 register. In addition to the main difference of being among a group of four parallel sets of locking registers, the other difference between the PASS_LOCK*x*_PG*n* registers and the flash LOCK*n* registers is how the reset values are determined.

By default, all mapped bits are set to ‘1’, locking their corresponding flash blocks from programming and erase. The reset value is user-configurable, though, by writing a DCF record. This is discussed in a later section.

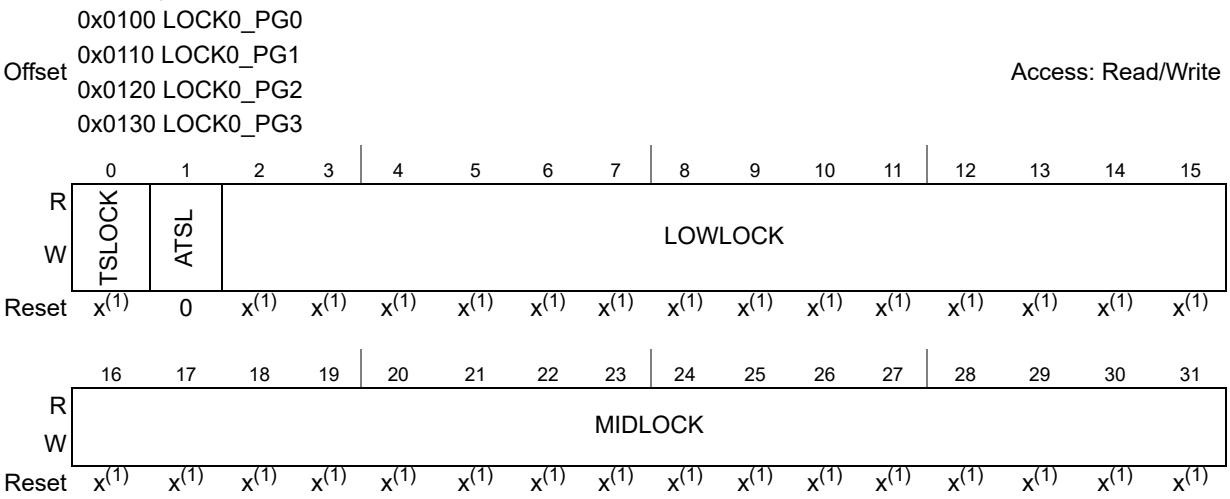


Figure 291. LOCK0_PG*n* register

1. If no DCF records have been written to change the lock state, the bit will reset to '1'

Note: The values in the PASS_LOCKxPGn registers override the settings in the flash module's LOCK0–3 registers.

Values written to the PASS_LOCKxPGn registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated flash block to be locked against programming or erase. The purpose of the registers is to indicate secure write protection status and to provide a way to temporarily override secure write protection.

The PASS_LOCKxPGn bits take effect after the life cycle is matured to "OEM Production" or older.

Example 15. Overriding secure write protection

Assume four passwords have been created but no DCF records have been written to set the reset values of the PASS_LOCK*x*_PG*n* registers, so the reset value for each mapped bit defaults to '1', causing all mapped blocks to have four levels of 256-bit password-secured write protection.

Since the settings in the PASS_LOCK*x*_PG*n* registers override the settings in the flash module's LOCK0–3 registers, the settings in those registers are irrelevant. Each of the four sets of the PASS module's LOCK registers must be unlocked by supplying the correct password for each set and then the appropriate bits must be set to '0' to make the desired flash blocks available for program or erase.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS CIN0.
3. Write 0xFFCF_FFFF to the PASS_LOCK0_PG0 register and verify a successful write.
4. Repeat the above steps for passwords 1–3.

Just as in previous examples, the result is the block Start Address: 0x00FC_C000 to End Address: 0x00FC_FFFF and block Start Address: 0x0060_C000 to End Address:

0x0060_FFFF are available for program and erase operations but at a much higher level of security.

31.4 Secure read protection

Read Protection is ability of the flash controller to block read accesses to the flash memory from all bus masters. This includes the blocking of:

- Instruction fetches and load instructions from the CPU
- Debug reads from an outside tool
- DMA reads
- SIPI reads
- All other types of bus master reads

When secure read protection is active, the affected flash blocks are readable by any bus master until a debugger is attached to the JTAG port and debug is enabled. If a debugger is attached and debug is enabled while an application is running, the next access to a flash read protected region terminates with a bus error.

Most instances of flash memory read protection in the SPC572Lx microcontroller are *not* configurable by the customer. An example of this is the 256-bit passwords stored in UTEST flash memory. Password-based read protection *is* customer-configurable[j] and is similar to the password-based write protection mechanism detailed in *Section 31.3: Secure write protection*.

There are two significant differences between password-based read protection and password-based write protection in the SPC572Lx microcontroller.

1. Read protection can only be applied to pre-defined groups of flash memory blocks. These groups, called “Read Lock groups”, cannot be re-defined, meaning the blocks assigned to those groups are permanently assigned and cannot be assigned to other groups and no new groups can be created.
2. Read protection is only active either when a debugger is attached to the JTAG port and debug is enabled; or the lifecycle has been advanced to “Failure Analysis”.

Another difference is that only the PASS_LOCK3_PG*n* register for each password group is used:

- PASS_LOCK3_PG0
- PASS_LOCK3_PG1
- PASS_LOCK3_PG2
- PASS_LOCK3_PG3

Note: The mappings of PASS_LOCK3_PGn register bits to flash blocks vary with different microcontrollers. See the flash memory section in the Device Configuration chapter of this reference manual for the mappings specific to the SPC572Lx microcontroller.

j. Secure read protection is user-configurable in that the default lock/unlock state for each mapped block at reset is controlled by user-created DCF records, giving the user the ability to determine the number of levels of password protection implemented. The read protection is applied to groups of blocks instead of individual blocks and the groups are not user-definable.

The next sections discuss implementing secure read protection and overriding secure write protection.

31.4.1 Implementing secure read protection

Implementation of secure read protection is done during initial flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) are stored in a protected OTP area of device flash named UTEST flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n*) that have the same mapping of register bits to flash blocks as the flash module's SEL0–3 registers and LOCK0–3 registers. These are the same passwords used by secure write protection and other secure functions. The passwords are only created once.
2. DCF records must be created to define which flash memory blocks are to be locked. Those records determine the reset values of the PASS_LOCK3_PG*n* registers.

   Like the other PASS_LOCK*x*_PG*n* registers, the PASS_LOCK3_PG*n* registers have two functions.

   – Reading the registers provides a lock status for each Read Locking group.

   – Changing the value of a mapped bit in a register changes the read protect status of the corresponding Read Locking group.

The layout of the PASS_LOCK3_PG*n* registers is shown in *Figure 292*. The fields relevant to implementing secure read protection are RL2–RL0.

Offset: 0x010C LOCK3_PG0, 0x011C LOCK3_PG1, 0x012C LOCK3_PG2, 0x013C LOCK3_PG3 — Access: Read/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	RL3	RL2	RL1	RL0
W																
Reset	x(1)	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	256LCK_U															
W																
Reset	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)

Figure 292. PASS_LOCK3_PG*n* Register

1. Reset value depends on life cycle. If life cycle is Customer delivery or earlier, the value will be 0, otherwise 1.
2. If no DCF records have been written to change the lock state, the bit will reset to '1'.

Example 16. Implementing secure read protection

Assume four passwords have been previously implemented.

Following is an example of a mapping of flash blocks to Read Locking groups.

Table 327. Sample low and mid address space flash block bit mapping

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Read Lock group	Block No.	Description
0x0040_4000	0x0040_7FFF	16	0	LOCK0	LOWLOCK[0]	16	—	0	BAF
0x00FC_0000	0x00FC_3FFF	16	0	LOCK0	LOWLOCK[1]	17	1	1	Code Flash
0x00FC_4000	0x00FC_7FFF	16	0	LOCK0	LOWLOCK[2]	18	1	2	Code Flash
0x00FC_8000	0x00FC_BFFF	16	1	LOCK0	LOWLOCK[3]	19	1	3	Code Flash
0x00FC_C000	0x00FC_FFFF	16	1	LOCK0	LOWLOCK[4]	20	1	4	Code Flash
0x00FD_0000	0x00FD_7FFF	32	0	LOCK0	LOWLOCK[6]	22	1	6	Code Flash
0x00FD_8000	0x00FD_FFFF	32	1	LOCK0	LOWLOCK[7]	23	1	7	Code Flash
0x00FE_0000	0x00FE_FFFF	64	0	LOCK0	LOWLOCK[8]	24	1	8	Code Flash
0x00FF_0000	0x00FF_FFFF	64	0	LOCK0	LOWLOCK[9]	25	1	9	Code Flash
0x0040_0000	0x0040_3FFF	16	0	LOCK0	TSLOCK	31	0	32	UTEST

To provide two level of password protection for only Read Lock group 3 and leave all other groups unlocked, the RL3 bit can be set in any two of the four PASS_LOCK3_PG*n* registers. For this example, we will choose password groups 0 and 1.

We need to create DCF records that define the reset values for all RL*n* bits as 0, with the exception of PASS_LOCK3_PG0 and PASS_LOCK3_PG1. The information required is from the PASS section of *Table 71: DCF client list* in *Chapter 8: Device Configuration Format (DCF) Records*.

Table 328. DCF client list of PASS

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0011	LOCK3_PG0
000_0000_0000_1000	000_0000_0100_0111	LOCK3_PG1
000_0000_0000_1000	000_0000_0100_1011	LOCK3_PG2
000_0000_0000_1000	000_0000_0100_1111	LOCK3_PG3

Referring to *Figure 289*, the values for the DCF record are:

- For LOCK3_PG0
  - Data[31:0] is 0x0008_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_0011 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG1
  - Data[31:0] is 0x0008_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_0111 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG2
  - Data[31:0] is 0x0000_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_1011 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG3
  - Data[31:0] is 0x0000_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_1111 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.

31.4.2 Overriding secure read protection

As noted previously, the PASS_LOCK3_PG*n* registers (see *Figure 292*) contain five read protection fields, RP0–RP4, that correspond to pre-defined Read Lock groups. To temporarily unlock a Read Lock group, the appropriate PASS_LOCK3_PG*n*[RL*n*] bits must be set to 0.

Like the register bits controlling secure write protection, the reset value of bits controlling secure read protection is determined by DCF record or, in the absence of a relevant DCF record, each bit resets to ‘1’, causing the blocks associated with each read group to be read locked.

Example 17. Overriding secure read protection

Building on *Example 16*, where Read Lock group 3 was protected by passwords 0 and 1, temporarily overriding secure read protection for that group is accomplished by first unlocking the PASS_LOCK3_PG0 and PASS_LOCK3_PG1 registers and then setting the RL3 bit to ‘0’ in both.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS CIN0.
3. Write a 0 to PASS_LOCK3_PG0[RL3] bit and verify a successful write.
4. Repeat the above steps for password 1.

The result of a successful sequence is that the blocks associated with the unlocked read lock group can be read by a debugger.

31.5 Debug port enable/disable

The mechanism for temporarily enabling the SPC572Lx debug port after it has been locked by its life cycle state is similar to the flash memory read protection mechanism detailed in *Section 31.4: Secure read protection*. The difference is that the PASS_LOCK3_PG*n*[DBL] field is used instead of the RP0–RP4 fields.

31.6 Implementing OTP

Any flash block within the flash array can be assigned, at any time, to be OTP. OTP means that flash erase of the entire block is disabled and only 64-bit double words that are already erased (i.e., flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block becomes OTP after the next reset.

Warning: After a flash block is configured as OTP, it cannot be changed back.

This capability resides within the Tamper Detection Module (TDM). The records are detailed in *Table 71: DCF client list* in *Chapter 8: Device Configuration Format (DCF) Records*. See the entries for the OTP_EN0, OTP_EN1, OTP_EN2, and OTP_EN3 DCF records in the Tamper Detect section of the table.

31.7 Implementing test mode disable

The SPC572Lx flash memory module includes a mechanism to disable manufacturer entry into test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method are able to have possible failures analyzed by manufacturer's failure analysts.

Test mode disable prevents all high voltage operations to the flash executed by the internal state-machine, as well as reads through the state machine, and reads through the Array Integrity state machine when using test mode interfaces.

Optionally, the customer can create a password to enable manufacturer entry into test mode.

31.7.1 Unconditional test mode disable seal

Warning: This process is not reversible. After completing these steps the SPC572Lx microcontroller is permanently prevented from entering test mode for factory analysis.

The steps for unconditionally disabling test mode are the same steps required for implementing passcode-protected entry into test mode with one exception—the passcode specified must be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To unconditionally disable test mode entry:

1. Select individual flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTEST flash memory map.
2. Program one of the three *invalid* passcodes shown above into the Test Mode Override Passcode area shown in the UTEST flash memory map.
3. Program the value 0x2D3C_4B5A into the Test Mode Disable Seal location specified in the UTEST flash memory map.

After the next reset is asserted, Test Mode is permanently disabled.

Note: Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.

The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

31.7.2 Passcode-protected test mode disable seal

The steps for implementing passcode-protected entry into test mode are the same required for unconditionally disabling test mode with the exception that —the passcode specified must NOT be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To implement passcode-protected test mode entry:

1. Select individual flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTEST flash memory map.
2. Program a valid passcode into the Test Mode Override Passcode area shown in the UTEST flash memory map.
3. Program the value 0x2D3C_4B5A into the Test Mode Disable Seal location specified in the UTEST flash memory map.

After the next reset is asserted, Test Mode is permanently passcode protected.

Note: Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.

The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

31.7.3 Selecting flash memory blocks for test mode disable seal

Selecting flash memory blocks for test mode disable seal protection involves writing to non-register flash locations. In the case of the test mode disable seal function, there are two areas of UTEST flash memory that have identical mapping to flash memory blocks. When the test mode disable seal takes effect, the mapped bits are logically ORed to determine the blocks to be protected.

The mapping of bits to blocks for this function is identical to the mapping used for other functions. See the table in the device configuration chapter for the map.

Note: Selecting a block for test mode disable requires a ‘0’ in the mapped bit—not a ‘1’ as with other functions.

31.8 Security configuration planning

31.8.1 Creating password groups

Most basic security features in the SPC572Lx microcontroller are password-based. Therefore, one of the early tasks that must be performed is the creation of four 256-bit passwords. They are programmed by the customer into the locations identified in the UTEST memory map (*Table 11*) in *Chapter 5: Memory map*.

The UTEST memory map fields to be written are:

- PASS Password Group 0
- PASS Password Group 1
- PASS Password Group 2
- PASS Password Group 3

Passwords must not be one of the following, which are considered insecure and therefore invalid:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

Warning: **The password fields are in OTP UTEST flash, so they can only be written once.**

31.8.2 Planning secure write protection

Up to four levels of password-secured write protection can be assigned to any block. Note that by default, all blocks of user flash are write protected, so this could be more accurately viewed as planning which blocks of user flash will *not* have four levels of password-secured write protection. The write protection desired will depend on the usage of the flash blocks. Before creating DCF records to implement secure write protection, thoroughly review the implementation steps detailed in *Section 31.3.1: Implementing secure write protection* and *Table 71: DCF client list* in *Chapter 8: Device Configuration Format (DCF) Records*.

31.8.3 Planning secure read protection

Up to four levels of password-secured write protection can be assigned to flash blocks in pre-defined Read Locking groups. Note that by default, all blocks of user flash are read protected, so this could be more accurately viewed as planning which blocks of user flash will *not* have four levels of password-secured read protection. The read protection desired will depend on the usage of the flash blocks. Before creating DCF records to implement secure read protection, thoroughly review the implementation steps detailed in *Section 31.4.1: Implementing secure read protection* and *Table 71: DCF client list* in *Chapter 8: Device Configuration Format (DCF) Records*.

31.8.4 Planning debug port enable/disable

See *Section 31.5: Debug port enable/disable* and *Table 71: DCF client list* in *Chapter 8: Device Configuration Format (DCF) Records*.

31.8.5 Planning OTP flash memory block assignment

Any block of user flash can be configured as OTP. The need to configure blocks as OTP will depend on the data or code to be stored in the block. It is important to note that configuring a flash block as OTP is not reversible.

Before creating DCF records to implemented OTP flash blocks, thoroughly review the implementation steps detailed in *Section 31.6: Implementing OTP* and *Table 71: DCF client list* in *Chapter 8: Device Configuration Format (DCF) Records*.

31.8.6 Planning factory test mode disable

See *Section 31.7: Implementing test mode disable* and *Table 71: DCF client list* in *Chapter 8: Device Configuration Format (DCF) Records*.

32 Error Reporting Module (ERM)

32.1 Introduction

32.1.1 Overview

The Error Reporting Module (ERM) provides information and optional interrupt notification on memory errors events associated with ECC (Error Correction Code). The ERM collects ECC events on memory accesses for platform local memory arrays, such as flash, system RAM or peripheral RAMs. Refer to the device configuration chapter for details on which memory sources are supported by the ERM and specific memory channel assignments.

32.1.2 Features

The ERM includes these features:

- capture address and syndrome information on single-bit correction and non-correctable ECC events
- optional interrupt notification on captured ECC events
- support for ECC event capturing for memory sources, with individual reporting fields and interrupt configuration per memory channel.

32.2 Memory map and register definition

The ERM 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 or in user mode generates an IPS error termination. The ERM 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 ERM module is in the middle of an operation will result in non-deterministic behavior.

32.2.1 Memory map

Table 329. ERM memory map

Address offset	Register	Access	Reset value	Location
0x000	ERM Configuration Register (ERM_CR)	R/W	0x0000_0000	*on page 626*
0x010	ERM Status Register (ERM_SR)	R/W	0x0000_0000	*on page 628*
0x100	ERM Memory 0 Error Address Register (ERM_EAR0)	R	0x0000_0000	*on page 632*
0x104	ERM Memory 0 Syndrome Register (ERM_SYN0)	R	0x0000_0000	*on page 632*
0x110	ERM Memory 1 Error Address Register (ERM_EAR1)	R	0x0000_0000	*on page 632*
0x114	ERM Memory 1 Syndrome Register (ERM_SYN1)	R	0x0000_0000	*on page 632*
0x120	ERM Memory 2 Error Address Register (ERM_EAR2)	R	0x0000_0000	*on page 632*

Table 329. ERM memory map(Continued)

Address offset	Register	Access	Reset value	Location
0x124	ERM Memory 2 Syndrome Register (ERM_SYN2)	R	0x0000_0000	on page 632
0x130	ERM Memory 3 Error Address Register (ERM_EAR3)	R	0x0000_0000	on page 632
0x134	ERM Memory 3 Syndrome Register (ERM_SYN3)	R	0x0000_0000	on page 632
0x140	ERM Memory 4 Error Address Register (ERM_EAR4)	R	0x0000_0000	on page 632
0x144	ERM Memory 4 Syndrome Register (ERM_SYN4)	R	0x0000_0000	on page 632
0x150	ERM Memory 5 Error Address Register (ERM_EAR5)	R	0x0000_0000	on page 632
0x154	ERM Memory 5 Syndrome Register (ERM_SYN5)	R	0x0000_0000	on page 632
0x160	ERM Memory 6 Error Address Register (ERM_EAR6)	R	0x0000_0000	on page 632
0x164	ERM Memory 6 Syndrome Register (ERM_SYN6)	R	0x0000_0000	on page 632
0x170	ERM Memory 7 Error Address Register (ERM_EAR7)	R	0x0000_0000	on page 632
0x174	ERM Memory 7 Syndrome Register (ERM_SYN7)	R	0x0000_0000	on page 632

32.2.2 Register descriptions

32.2.2.1 ERM Configuration Register (ERM_CR)

This ERM Configuration Register is a 32-bit control register for configuring the interrupt notification capability for memory channels 0 through 7.

Offset 0x000

Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ESCIE0	ENCIE0	0	0	ESCIE1	ENCIE1	0	0	ESCIE2	ENCIE2	0	0	ESCIE3	ENCIE3	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ESCIE4	ENCIE4	0	0	ESCIE5	ENCIE5	0	0	ESCIE6	ESCIE6	0	0	ESCIE7	ESCIE7	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 293. ERM Configuration Register (ERM_CR)

Table 330. ERM_CR field descriptions

Field	Description
0 ESCIE0	Enable Memory 0 Single Correction Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 0 single-bit correction events is disabled. 1 Interrupt notification of Memory 0 single-bit correction events is enabled. Refer to the device configuration chapter for details on Memory 0 mapping.
1 ENCIE0	Enable Memory 0 Non-Correctable Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 0 non-correctable error events is disabled. 1 Interrupt notification of Memory 0 non-correctable error events is enabled. Refer to the device configuration chapter for details on Memory 0 mapping.
4 ESCIE1	Enable Memory 1 Single Correction Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 1 single-bit correction events is disabled. 1 Interrupt notification of Memory 1 single-bit correction events is enabled. Refer to the device configuration chapter for details on Memory 1 mapping.
5 ENCIE1	Enable Memory 1 Non-Correctable Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 1 non-correctable error events is disabled. 1 Interrupt notification of Memory 1 non-correctable error events is enabled. Refer to the device configuration chapter for details on Memory 1 mapping.
8 ESCIE2	Enable Memory 2 Single Correction Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 2 single-bit correction events is disabled. 1 Interrupt notification of Memory 2 single-bit correction events is enabled. Refer to the device configuration chapter for details on Memory 2 mapping.
9 ENCIE2	Enable Memory 2 Non-Correctable Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 2 non-correctable error events is disabled. 1 Interrupt notification of Memory 2 non-correctable error events is enabled. Refer to the device configuration chapter for details on Memory 2 mapping.
12 ESCIE3	Enable Memory 3 Single Correction Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 3 single-bit correction events is disabled. 1 Interrupt notification of Memory 3 single-bit correction events is enabled. Refer to the device configuration chapter for details on Memory 3 mapping.
13 ENCIE3	Enable Memory 3 Non-Correctable Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 3 non-correctable error events is disabled. 1 Interrupt notification of Memory 3 non-correctable error events is enabled. Refer to the device configuration chapter for details on Memory 3 mapping.
16 ESCIE4	Enable Memory 4 Single Correction Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 4 single-bit correction events is disabled. 1 Interrupt notification of Memory 4 single-bit correction events is enabled. Refer to the device configuration chapter for details on Memory 4 mapping.

Table 330. ERM_CR field descriptions(Continued)

Field	Description
17 ENCIE4	Enable Memory 4 Non-Correctable Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 4 non-correctable error events is disabled. 1 Interrupt notification of Memory 4 non-correctable error events is enabled. Refer to the device configuration chapter for details on Memory 4 mapping.
20 ESCIE5	Enable Memory 5 Single Correction Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 5 single-bit correction events is disabled. 1 Interrupt notification of Memory 5 single-bit correction events is enabled. Refer to the device configuration chapter for details on Memory 5 mapping.
21 ENCIE5	Enable Memory 5 Non-Correctable Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 5 non-correctable error events is disabled. 1 Interrupt notification of Memory 5 non-correctable error events is enabled. Refer to the device configuration chapter for details on Memory 5 mapping.
24 ESCIE6	Enable Memory 6 Single Correction Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 6 single-bit correction events is disabled. 1 Interrupt notification of Memory 6 single-bit correction events is enabled. Refer to the device configuration chapter for details on Memory 6 mapping.
25 ENCIE6	Enable Memory 6 Non-Correctable Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 6 non-correctable error events is disabled. 1 Interrupt notification of Memory 6 non-correctable error events is enabled. Refer to the device configuration chapter for details on Memory 6 mapping.
28 ESCIE7	Enable Memory 7 Single Correction Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 7 single-bit correction events is disabled. 1 Interrupt notification of Memory 7 single-bit correction events is enabled. Refer to the device configuration chapter for details on Memory 7 mapping.
29 ENCIE7	Enable Memory 7 Non-Correctable Interrupt Notification This field is initialized by hardware reset. 0 Interrupt notification of Memory 7 non-correctable error events is disabled. 1 Interrupt notification of Memory 7 non-correctable error events is enabled. Refer to the device configuration chapter for details on Memory 7 mapping.

32.2.2.2 ERM Status Register (ERM_SR)

This ERM Status Register is a 32-bit control register for signaling which types of ECC events have been detected for memory channels 0 through 7. The ERM_SR signals the last memory event to be detected for each memory channel.

Offset 0x010

Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SBC0	NCE0	0	0	SBC1	NCE1	0	0	SBC2	NCE2	0	0	SBC3	NCE3	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SBC4	NCE4	0	0	SBC5	NCE5	0	0	SBC6	NCE6	0	0	SBC7	NCE7	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 294. ERM Status Register (ERM_SR)

Table 331. ERM_SR field descriptions

Field	Description
0 SBC0	Memory 0 Single-Bit Correction Event This field is initialized by hardware reset. Write ‘1’ to clear this field. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ESCIE0] is enabled. 0 No single-bit correction event on Memory 0 detected 1 Single-bit correction event on Memory 0 detected **Note:** Refer to the device configuration chapter for details on Memory 0 mapping.
1 NCE0	Memory 0 Non-Correctable Error Event This field is initialized by hardware reset. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ENCIE0] is enabled. 0 No non-correctable error event on Memory 0 detected 1 Non-correctable error event on Memory detected **Note:** Refer to the device configuration chapter for details on Memory 0 mapping.
4 SBC1	Memory 1 Single-Bit Correction Event This field is initialized by hardware reset. Write ‘1’ to clear this field. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ESCIE1] is enabled. 0 No single-bit correction event on Memory 1 detected 1 Single-bit correction event on Memory 1 detected **Note:** Refer to the device configuration chapter for details on Memory 1 mapping.
5 NCE1	Memory 1 Non-Correctable Error Event This field is initialized by hardware reset. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ENCIE1] is enabled. 0 No non-correctable error event on Memory 1 detected 1 Non-correctable error event on Memory 1 detected **Note:** Refer to the device configuration chapter for details on Memory 1 mapping.

Table 331. ERM_SR field descriptions(Continued)

Field	Description
8 SBC2	Memory 2 Single-Bit Correction Event This field is initialized by hardware reset. Write ‘1’ to clear this field. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ESCIE2] is enabled. 0 No single-bit correction event on Memory 2 detected 1 Single-bit correction event on Memory 2 detected **Note:** Refer to the device configuration chapter for details on Memory 2 mapping.
9 NCE2	Memory 2 Non-Correctable Error Event This field is initialized by hardware reset. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ENCIE2] is enabled. 0 No non-correctable error event on Memory 2 detected 1 Non-correctable error event on Memory 2 detected **Note:** Refer to the device configuration chapter for details on Memory 2 mapping.
12 SBC3	Memory 12 Single-Bit Correction Event This field is initialized by hardware reset. Write ‘1’ to clear this field. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ESCIE3] is enabled. 0 No single-bit correction event on Memory 3 detected 1 Single-bit correction event on Memory 3 detected **Note:** Refer to the device configuration chapter for details on Memory 3 mapping.
13 NCE3	Memory 3 Non-Correctable Error Event This field is initialized by hardware reset. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ENCIE3] is enabled. 0 No non-correctable error event on Memory 3 detected 1 Non-correctable error event on Memory 3 detected **Note:** Refer to the device configuration chapter for details on Memory 3 mapping.
16 SBC4	Memory 4 Single-Bit Correction Event This field is initialized by hardware reset. Write ‘1’ to clear this field. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ESCIE4] is enabled. 0 No single-bit correction event on Memory 4 detected 1 Single-bit correction event on Memory 4 detected **Note:** Refer to the device configuration chapter for details on Memory 4 mapping.
17 NCE4	Memory 4 Non-Correctable Error Event This field is initialized by hardware reset. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ENCIE4] is enabled. 0 No non-correctable error event on Memory 4 detected 1 Non-correctable error event on Memory 4 detected **Note:** Refer to the device configuration chapter for details on Memory 4 mapping.

Table 331. ERM_SR field descriptions(Continued)

Field	Description
20 SBC5	Memory 5 Single-Bit Correction Event This field is initialized by hardware reset. Write ‘1’ to clear this field. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ESCIE5] is enabled. 0 No single-bit correction event on Memory 5 detected 1 Single-bit correction event on Memory 5 detected **Note:** Refer to the device configuration chapter for details on Memory 5 mapping.
21 NCE5	Memory 5 Non-Correctable Error Event This field is initialized by hardware reset. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ENCIE5] is enabled. 0 No non-correctable error event on Memory 5 detected 1 Non-correctable error event on Memory 5 detected **Note:** Refer to the device configuration chapter for details on Memory 5 mapping.
24 SBC6	Memory 6 Single-Bit Correction Event This field is initialized by hardware reset. Write ‘1’ to clear this field. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ESCIE6] is enabled. 0 No single-bit correction event on Memory 6 detected 1 Single-bit correction event on Memory 6 detected **Note:** Refer to the device configuration chapter for details on Memory 6 mapping.
25 NCE6	Memory 6 Non-Correctable Error Event This field is initialized by hardware reset. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ENCIE6] is enabled. 0 No non-correctable error event on Memory 6 detected 1 Non-correctable error event on Memory 6 detected **Note:** Refer to the device configuration chapter for details on Memory 6 mapping.
28 SBC7	Memory 7 Single-Bit Correction Event This field is initialized by hardware reset. Write ‘1’ to clear this field. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ESCIE7] is enabled. 0 No single-bit correction event on Memory 7 detected 1 Single-bit correction event on Memory 7 detected **Note:** Refer to the device configuration chapter for details on Memory 7 mapping.
29 NCE7	Memory 7 Non-Correctable Error Event This field is initialized by hardware reset. Write ‘1’ to clear this field also deasserts the corresponding interrupt notification if ERM_CR[ENCIE7] is enabled. 0 No non-correctable error event on Memory 7 has been not been detected 1Non-correctable error event on Memory 7 detected **Note:** Refer to the device configuration chapter for details on Memory 7 mapping.

32.2.2.3 ERM Memory *n* Error Address Register (ERM_EAR*n*)

The ERM_EARn is a 32-bit register for capturing the address of the last ECC event in Memory *n*, where *n* denotes the memory channel. Any attempted write to ERM_EARn is ignored.

Offset 0x100 + (0x010*n) + 0x00

Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EAR[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	EAR[0:15]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 295. ERM Memory n Error Address Register (ERM_EARn)

Table 332. ERM_EARn field description

Field	Description
0-31 EAR	Memory *n* Error Address – This field contains the faulting system address of the last recorded ECC event on Memory *n*. **Note:** Refer to the device configuration chapter for details on Memory *n* mapping.

32.2.2.4 ERM Memory *n* Syndrome Register (ERM_SYN*n*)

The ERM_SYN*n* is a 32-bit register for capturing the calculated syndrome of the last ECC event on Memory *n,* where *n* denotes the memory channel. Any attempted write to ERM_SYN*n* is ignored. The syndrome value identifies the pertinent bit position on a correctable, single-bit data inversion or a non-correctable, single-bit address inversion. The syndrome value does not provide any additional diagnostic information on non-correctable, multi-bit inversions.

Offset 0x100 + (0x010*n) + 0x04

Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYNDROME[0:7]								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	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 296. ERM Memory n Syndrome Register (ERM_SYNn)

Table 333. ERM_SYNn field description

Field	Description
0-7 SYNDROME	Memory n Syndrome - This field contains the ECC syndrome associated with the last recorded ECC event on Memory *n*. **Note:** Refer to the device configuration chapter for details on Memory *n* mapping.

32.3 Functional description

32.3.1 Single-bit correction events

Single-bit correction events reported from Memory *n* are recorded in the Error Reporting Module Status Register (ERM_SR[SBCn]). The corresponding access address which initiated the ECC event is stored in the Memory *n* Error Address Register (ERM_EAR*n*). The corresponding ECC syndrome is stored in the Memory *n* Error Syndrome Register (ERM_SYN*n*), which identifies the bit position of the corrected data on single-bit data inversion. The ERM can be optionally configured to generate an interrupt notification upon reporting of a single-bit ECC correction on Memory *n* by enabling ERM_ECR[SBCIE*n*]. Writing a ‘1’ to ERM_SR[SBC*n*] clears this field and deasserts the interrupt notification if interrupt notification is enabled for this memory channel.

The ERM only holds event information for the last reported event.

32.3.2 Non-correctable error events

Non-correctable ECC error events reported from Memory *n* are recorded in the Error Reporting Module Status Register (ERM_SR[NCE*n*]). The corresponding access address which initiated the ECC event is stored in the Memory *n* Error Address Register (ERM_EAR*n*). The corresponding ECC syndrome is stored in the Memory *n* Error Syndrome Register (ERM_SYN*n*). In the event of a non-correctable address bit inversion, the ERM_SYN*n* register identifies the identified address bit position. In the event of a non-correctable, multi-bit data inversion, the syndrome value does not provide any additional diagnostic information.

The ERM only holds event information for the last reported event.

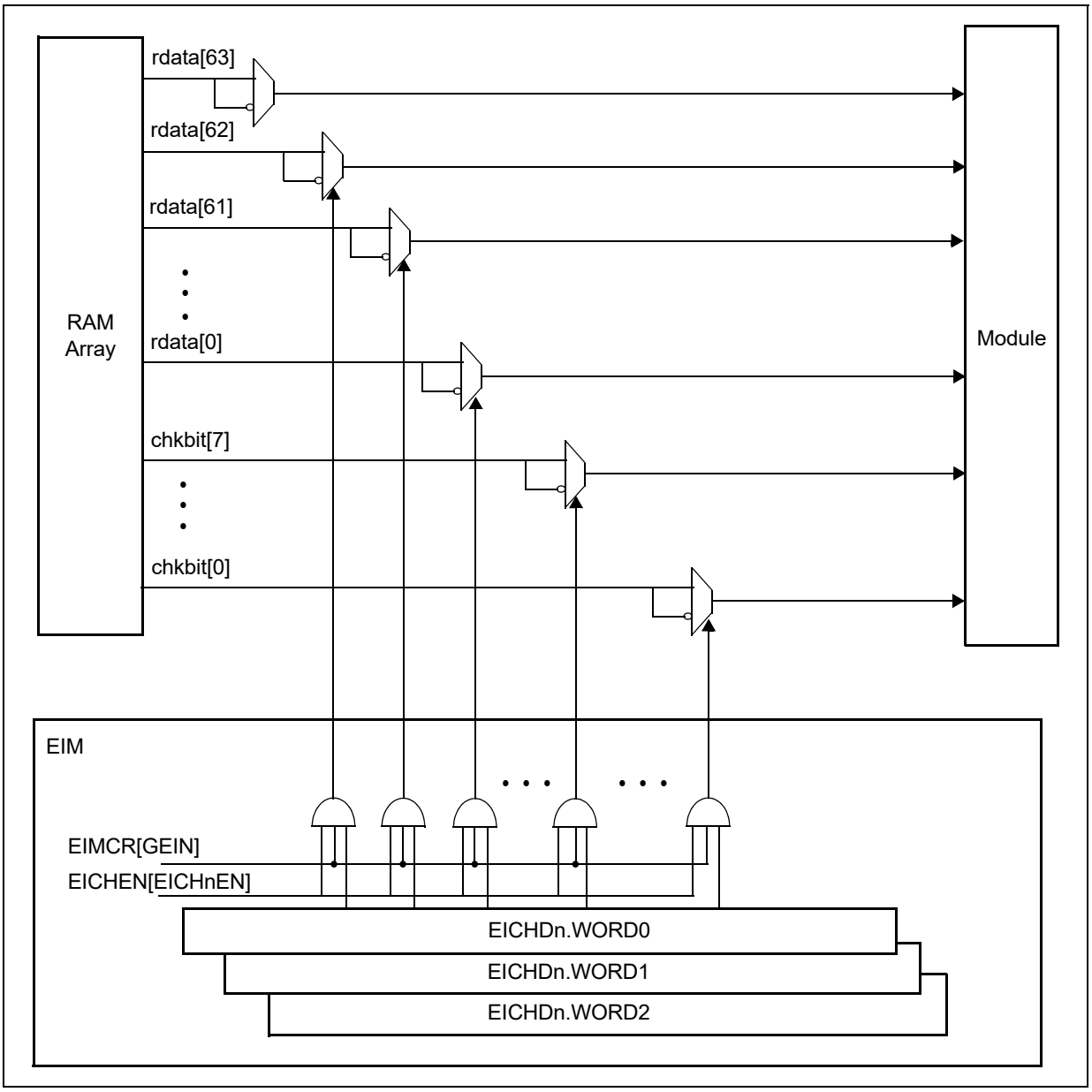
33 Error Injection Module (EIM)

33.1 Introduction

33.1.1 Overview

The Error Injection Module (EIM) provides support for inducing single-bit and multi-bit inversions on read data when accessing peripheral RAMs. Injecting errors on memory accesses can be used to exercise the SEC/DED ECC function of the related system.

Figure 297. EIM functional block diagram



33.1.2 Features

The EIM includes these features:

- supports up to six error injection channels
- protection against accidental enable and reconfiguration error injection function via two-stage enable mechanism

33.2 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 or in user mode generates an IPS error termination. The EIM 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 EIM module is in the midst of an operation will result in non-deterministic behavior.

33.2.1 Memory map

Table 334. EIM memory map

Address offset	Register	Access	Reset value	Section/Page
0x000	Error Injection Module Configuration Register (EIMCR)	R/W	0x0000_0000	*33.2.2.1/636*
0x004	Error Injection Channel Enable Register (EICHEN)	R/W	0x0000_0000	*33.2.2.2/637*
0x00A–0x0FF	Reserved			
0x100	Error Injection Channel0 Descriptor, Word0 (EICHD0.Word0)	R/W	0x0000_0000	*33.2.2.3.1/639*
0x104	Error Injection Channel0 Descriptor, Word1 (EICHD0.Word1)	R/W	0x0000_0000	*33.2.2.3.2/640*
0x108	Error Injection Channel0 Descriptor, Word2 (EICHD0.Word2)	R/W	0x0000_0000	*33.2.2.3.3/641*
0x10C–0x1FF	Reserved			
0x200	Error Injection Channel1 Descriptor, Word0 (EICHD1.Word0)	R/W	0x0000_0000	*33.2.2.3.1/639*
0x204	Error Injection Channel1 Descriptor, Word1 (EICHD1.Word1)	R/W	0x0000_0000	*33.2.2.3.2/640*
0x208	Error Injection Channel1 Descriptor, Word2 (EICHD1.Word2)	R/W	0x0000_0000	*33.2.2.3.3/641*
0x20C–0x2FF	Reserved			
0x300	Error Injection Channel2 Descriptor, Word0 (EICHD2.Word0)	R/W	0x0000_0000	*33.2.2.3.1/639*

Table 334. EIM memory map(Continued)

Address offset	Register	Access	Reset value	Section/Page
0x304	Error Injection Channel2 Descriptor, Word1 (EICHD2.Word1)	R/W	0x0000_0000	*33.2.2.3.2/640*
0x308	Error Injection Channel2 Descriptor, Word2 (EICHD2.Word2)	R/W	0x0000_0000	*33.2.2.3.3/641*
0x30C–0x3FF	Reserved			
0x400	Error Injection Channel3 Descriptor, Word0 (EICHD3.Word0)	R/W	0x0000_0000	*33.2.2.3.1/639*
0x404	Error Injection Channel3 Descriptor, Word1 (EICHD3.Word1)	R/W	0x0000_0000	*33.2.2.3.2/640*
0x408	Error Injection Channel3 Descriptor, Word2 (EICHD3.Word2)	R/W	0x0000_0000	*33.2.2.3.3/641*
0x40C–0x4FF	Reserved			
0x500	Error Injection Channel4 Descriptor, Word0 (EICHD4.Word0)	R/W	0x0000_0000	*33.2.2.3.1/639*
0x504	Error Injection Channel4 Descriptor, Word1 (EICHD4.Word1)	R/W	0x0000_0000	*33.2.2.3.2/640*
0x508	Error Injection Channel4 Descriptor, Word2 (EICHD4.Word2)	R/W	0x0000_0000	*33.2.2.3.3/641*
0x50C–0x5FF	Reserved			
0x600	Error Injection Channel5 Descriptor, Word0 (EICHD5.Word0)	R/W	0x0000_0000	*33.2.2.3.1/639*
0x604	Error Injection Channel5 Descriptor, Word1 (EICHD5.Word1)	R/W	0x0000_0000	*33.2.2.3.2/640*
0x608	Error Injection Channel5 Descriptor, Word2 (EICHD5.Word2)	R/W	0x0000_0000	*33.2.2.3.3/641*

33.2.2 Register descriptions

Following sections describe the registers associated with this module.

33.2.2.1 Error Injection Module Configuration Register (EIMCR)

The EIM Configuration Register is used to globally enable or disable the error injection function.

Offset 0x000

Access: Supervisor Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GEIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 298. Error Injection Module Configuration Register (EIMCR)

Table 335. Error Injection Module Configuration Register (EIMCR) field description

Field	Description
31 GEIEN	Global Error Injection Enable. This bit globally enables or disables the error injection function of the EIM. This field is initialized by hardware reset. 0 Error injection is not performed on any RAM read accesses. 1 Error injection is performed on RAM read accesses as configured in Error Injector Channels 0–5.

33.2.2.2 Error Injection Channel Enable (EICHEN) register

The Error Injection Channel Enable (EICHEN) register is used to enable or disable the individual error injection channels.

Note: There is a global error injection enable (EIMCR[GEIEN]) that must also be asserted in conjunction with the individual EICHnEN flags to enable a given error injection channel.

Offset 0x004

Access: Supervisor Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EICH0EN	EICH1EN	EICH2EN	EICH3EN	EICH4EN	EICH5EN	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	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 299. Error Injection Channel Enable (EICHEN) register

Table 336. Error Injection Channel Enable register (EICHEN) field description

Name	Description
31 EICH0EN	Error Injection Channel 0 Enable. This bit enables the corresponding error injection channel. There is global error injection enable (EIMCR[GEIEN]) also that must be asserted to enable error injection. Once error injection is enabled, all subsequent read accesses will incur bit inversion(s) as defined in Error Injection Descriptor0 registers (EICHD0) until the error injection channel is manually disabled via software. Any write to the corresponding EICHDn registers clears the corresponding EICHEN[EICHnEN] bit, leaving the error injection channel disabled. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Error injection is disabled on Error Injector Channel0 1 Error injection is enabled on Error Injector Channel0
30 EICH1EN	Error Injection Channel 1 Enable. This bit enables the corresponding error injection channel. There is a global error injection enable (EIMCR[GEIEN]) also that must be asserted to enable error injection. Once error injection is enabled, all subsequent read accesses will incur bit inversion(s) as defined in Error Injection Descriptor1 registers (EICHD1) until the error injection channel is manually disabled via software. Any write to the corresponding EICHDn registers clears the corresponding EICHEN[EICHnEN] bit, leaving the error injection channel disabled. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Error injection is disabled on Error Injector Channel1 1 Error injection is enabled on Error Injector Channel1
29 EICH2EN	Error Injection Channel 2 Enable. This bit enables the corresponding error injection channel. There is a global error injection enable (EIMCR[GEIEN]) also that must be asserted to enable error injection. Once error injection is enabled, all subsequent read accesses will incur bit inversion(s) as defined in Error Injection Descriptor2 registers (EICHD2) until the error injection channel is manually disabled via software. Any write to the corresponding EICHDn registers clears the corresponding EICHEN[EICHnEN] bit, leaving the error injection channel disabled. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Error injection is disabled on Error Injector Channel 2 1 Error injection is enabled on Error Injector Channel 2
28 EICH3EN	Error Injection Channel 3 Enable. This bit enables the corresponding error injection channel. There is a global error injection enable (EIMCR[GEIEN]) also that must be asserted to enable error injection. Once error injection is enabled, all subsequent read accesses will incur bit inversion(s) as defined in Error Injection Descriptor3 registers (EICHD3) until the error injection channel is manually disabled via software. Any write to the corresponding EICHDn registers clears the corresponding EICHEN[EICHnEN] bit, leaving the error injection channel disabled. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Error injection is disabled on Error Injector Channel 3 1 Error injection is enabled on Error Injector Channel 3

Table 336. Error Injection Channel Enable register (EICHEN) field description(Continued)

Name	Description
27 EICH4EN	Error Injection Channel 4 Enable. This bit enables the corresponding error injection channel. There is a global error injection enable (EIMCR[GEIEN]) also that must be asserted to enable error injection. Once error injection is enabled, all subsequent read accesses will incur bit inversion(s) as defined in Error Injection Descriptor4 registers (EICHD4) until the error injection channel is manually disabled via software. Any write to the corresponding EICHDn registers clears the corresponding EICHEN[EICHnEN] bit, leaving the error injection channel disabled. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Error injection is disabled on Error Injector Channel 4 1 Error injection is enabled on Error Injector Channel 4
26 EICH5EN	Error Injection Channel 5 Enable. This bit enables the corresponding error injection channel. There is a global error injection enable (EIMCR[GEIEN]) also that must be asserted to enable error injection. Once error injection is enabled, all subsequent read accesses will incur bit inversion(s) as defined in Error Injection Descriptor5 registers (EICHD5) until the error injection channel is manually disabled via software. Any write to the corresponding EICHDn registers clears the corresponding EICHEN[EICHnEN] bit, leaving the error injection channel disabled. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Error injection is disabled on Error Injector Channel 5 1 Error injection is enabled on Error Injector Channel 5

33.2.2.3 Error Injection Channel n Descriptor (EICHDn)

Each 96-bit (12-byte) error injection channel descriptor specifies a mask that defines which bits of the read data and checkbit bus from target RAM are inverted on a read access. The error injection channel descriptors are organized sequentially as 128-bit (16-byte) structures in the Error Injection Module programming model. Each of the three 32-bit words that define a single error injection channel are detailed in the subsequent sections; the fourth word is unused.

In the subsequent sections the terminology, *upper word*, is used to describe bits 63–32 of a 64-bit data bus.

In the subsequent sections the terminology, *lower word*, is used to describe bits 31–0 of a 64-bit data bus.

33.2.2.3.1 Error Injection Channel n Descriptor, Word0 (EICHDn.Word0)

The first word of the Error Injection Channel n Descriptor defines an 8-bit mask (CHKBIT_MASK). Each bit of CHKBIT_MASK specifies if the corresponding bit of the checkbit bus from the target RAM should be inverted or left unmodified on read accesses.

Successful writes to this word clear the corresponding error injection channel valid bit, EICHEN[EICHnEN].

Offset 0x100 + (0x100*n) + 0x00

Access: Supervisor Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CHKBIT_MASK								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	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 300. Error Injection Channel n Descriptor, Word0 (EICHDn.Word0)

Table 337. Error Injection Channel n Descriptor, Word0 (EICHDn.Word0) field description

Name	Description
31-24 CHKBIT_MASK	Checkbit Mask. This field defines a bit-mapped mask that specifies if the corresponding bit of the checkbit bus from the target RAM should be inverted or left unmodified. 0 The corresponding bit of the checkbit bus is left unmodified. 1 The corresponding bit of the checkbit bus is inverted.

33.2.2.3.2 Error Injection Channel n Descriptor, Word1 (EICHDn.Word1)

The second word of the Error Injection Channel n Descriptor defines the 32-bit mask (UDATA_MASK). Each bit of UDATA_MASK specifies if the corresponding bit of the upper word on the 64-bit read data bus from the target RAM should be inverted or left unmodified on read accesses. The upper word corresponds to bits 63–32 of the 64-bit read data bus.

Successful writes to this word clear the corresponding error injection channel valid bit, EICHEN[EICHnEN].

Offset 0x100 + (0x100*n) + 0x04

Access: Supervisor Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UDATA_MASK															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UDATA_MASK															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 301. Error Injection Channel n Descriptor, Word1 (EICHDn.Word1)

Table 338. Error Injection Channel n Descriptor, Word1 (EICHDn.Word1) field description

Name	Description
31-0 UDATA_MASK	Upper Data Mask. This field defines a bit-mapped mask that specifies if the corresponding bit of the upper word on the read data bus from the target RAM should be inverted or left unmodified. 0 The corresponding bit of the upper word on the read data bus is left unmodified. 1 The corresponding bit of the upper word on the read data bus is inverted.

33.2.2.3.3 Error Injection Channel n Descriptor, Word2 (EICHDn.Word2)

The third word of the Error Injection Channel n Descriptor defines a 32-bit mask (LDATA_MASK). Each bit of LDATA_MASK specifies if the corresponding bit of the lower word on the 64-bit read data bus from the target RAM should be inverted or left unmodified on read accesses.The lower word corresponds to bits 31–0 of the 64-bit read data bus.

Successful writes to this word clear the corresponding error injection channel valid bit, EICHEN[EICHnEN].

Offset 0x100 + (0x100*n) + 0x08

Access: Supervisor Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LDATA_MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LDATA_MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 302. Error Injection Channel n Descriptor, Word2 (EICHDn.Word2)

Table 339. Error Injection Channel n Descriptor, Word2 (EICHDn.Word2) Field Description

Name	Description
31-0 LDATA_MASK	Lower Data Mask. This field defines a bit-mapped mask that specifies if the corresponding bit of the lower word on the read data bus from the target RAM should be inverted or left unmodified. 0 The corresponding bit of the lower word on the read data bus is left unmodified. 1 The corresponding bit of the lower word on the read data bus is inverted.

33.3 Functional description

The EIM provides protection against accidental enable and reconfiguration of the error injection function via a two-stage enable mechanism. To properly enable the error injection

mechanism for any given channel, the following two-stage enable scheme must be executed:

- EICHEN[EICHnEN] field is asserted, where n denotes the channel number
- EIMCR[GEIEN] is asserted

The EIM supports up to 4 error injection channels, where each channel is assigned to a single memory array interface. Each EIM channel intercepts the assigned memory read data and checkbit bus and potentially inverts the value transmitted on each bus line as defined by EICHDn.WORD0-2. On a memory read access, the read data and checkbit bus is intercepted by the EIM and the appropriate bits of the read data and/or checkbit bus are inverted, injecting error(s), as defined by EICHDn.WORD0-2. *Figure 297* depicts the interception and override of the read data and checkbit data bus for a memory array.

34 Analog-to-digital converter (ADC) configuration

34.1 ADC overview

This chapter is organized into the following sections:

- *Section 34.1, ADC overview*
  - *Section 34.1.1, ADC subsystem block diagram*
  - *Section 34.1.2, Analog input pin multiplexing*
- *Section 34.2, Configuration of ADC modules*
  - *Section 34.2.1, Sigma-Delta Analog-to-Digital Converter (SDADC)*
  - *Section 34.2.2, Successive Approximation Register Analog-to-Digital Converter (SARADC)*

SPC572Lx contains the following analog-to-digital converters (ADCs):

- 3 independent 12-bit Successive Approximation Register Analog-to-Digital Converters (SARADCs)
- 1 independent 16-bit Sigma-Delta Analog-to-Digital Converter (Sigma-Delta, or SDADC)

These ADCs interface to 18 external analog input pins in the 80-pin QFP. The ADCs interface to 24 external analog input pins in the 100-pin QFP.

Each ADC instance, whether SAR or Sigma-Delta, has independent analog input pin multiplexing, register controls, trigger and clock inputs, independent interrupt and DMA capability. Each SAR and Sigma-Delta ADC has a dedicated peripheral bridge (PBRIDGE) interface for CPU access to the registers.

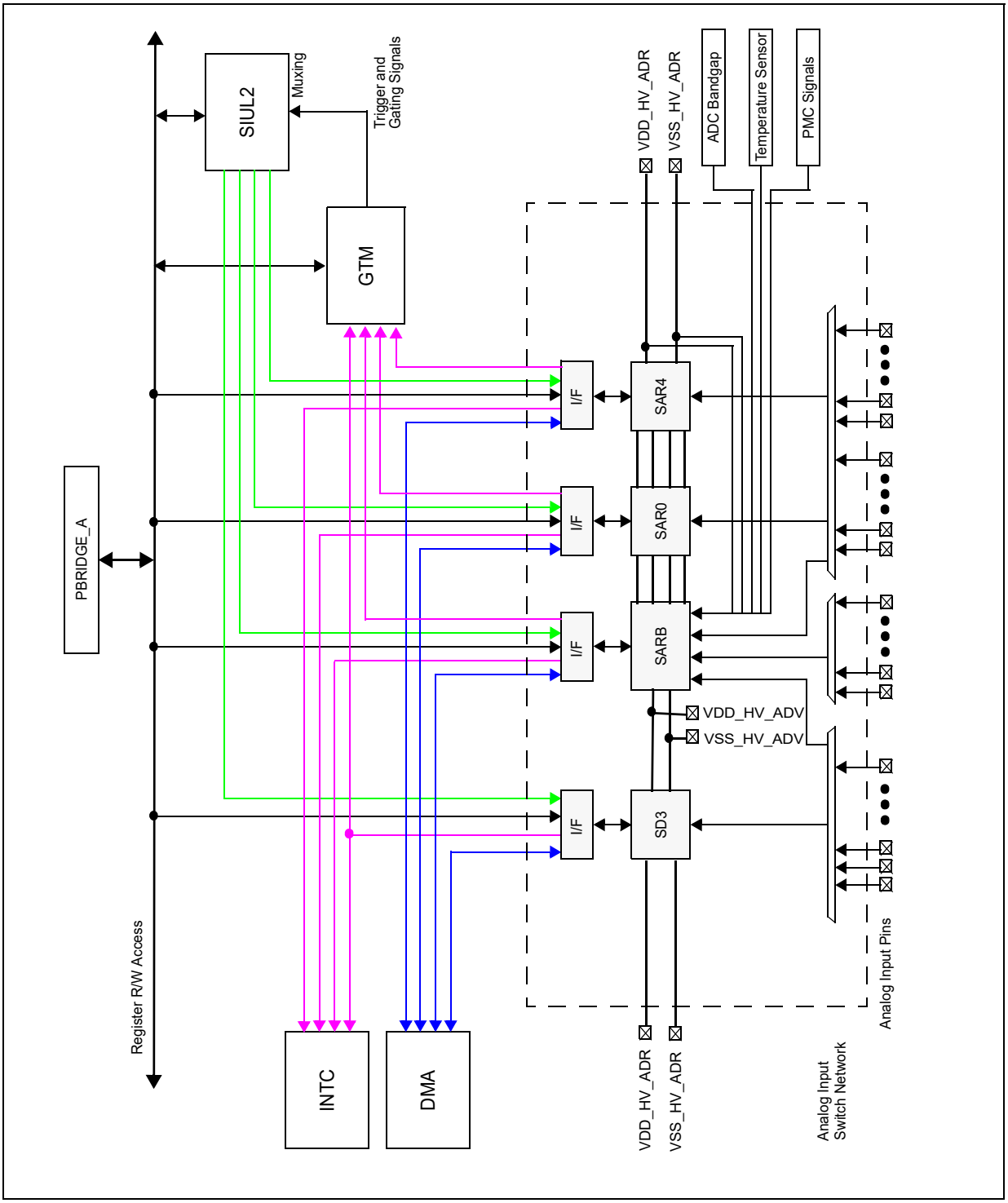
All SAR and Sigma-Delta ADCs share the same supply pins (VDD_HV_ADV/VSS_HV_ADV), ground, as well as reference pins. This is true for both packages.

Note that the numbering scheme is aligned along the complete family of devices. The SAR ADCs on the device are referred to as SARB (24:1 input mux), SAR0, and SAR4 (maximum 9:1 input mux for each).

34.1.1 ADC subsystem block diagram

Figure 303 gives the block diagram for the high-level integration of the ADC subsystem for SPC572Lx. The primary intent of the diagram is to show that each ADC, whether SAR or Sigma-Delta, has independent register interfaces, interrupt/DMA requests, and trigger source selection.

Figure 303. SPC572Lx ADC subsystem



34.1.2 Analog input pin multiplexing

There are 24 external analog input pins in the 100-pin QFP for SPC572Lx, and 18 in the 80-pin QFP. All analog input pins in either package are readable by SAR ADCB. In addition to

SAR ADCB, most analog input pins are routed to either the Sigma-Delta or SAR ADCs, but never both. *Table 340* gives the analog input pin assignment for both SPC572Lx 100-pin and 80-pin packages.

Table 340. SPC572Lx analog input pin multiplexing

ADC instance	80-pin QFP		100-pin QFP	
	Analog input pins	Mux input(1)	Analog input pins	Mux input(1)
SAR ADCB	AN[0:7] AN[16:17] AN[36:39] AN[58:61]	AN[0:7] AN[16:17] AN[36:39] AN[58:61]	AN[0:8] AN[16:17] AN[24] AN[36:39] AN[52:53] AN[56:61]	AN[0:8] AN[16:17] AN[24] AN[36:39] AN[52:53] AN[56:61]
SAR ADC0	AN[0:7]	AN[0:7]	AN[0:8]	AN[0:8]
SAR ADC4	AN[58:61]	AN[58:61]	AN[52:53] AN[56:61]	AN[52:53] AN[56:61]
SD ADC3	AN[36:37] AN[38:39]	AN[0:1] AN[2:3]	AN[36:37] AN[38:39]	AN[0:1] AN[2:3]

1. For the Sigma-Delta mux inputs AN[x], the positive and negative input terminals are selected in the Sigma-Delta ADC digital interface.

34.2 Configuration of ADC modules

The following sections describe device-specific details for the following modules:

- *Section 34.2.1, Sigma-Delta Analog-to-Digital Converter (SDADC)*
- *Section 34.2.2, Successive Approximation Register Analog-to-Digital Converter (SARADC)*

34.2.1 Sigma-Delta Analog-to-Digital Converter (SDADC)

There is one independent Sigma-Delta ADC on SPC572Lx. Refer to *Figure 303* for the ADC subsystem block diagram.

The following sections describe configuration details:

- *Section 34.2.1.1, Sigma-Delta ADC external signal description*
- *Section 34.2.1.2, Sigma-Delta ADC integration diagram*
- *Section 34.2.1.3, Sigma-Delta ADC analog input multiplexing*
- *Section 34.2.1.4, Sigma-Delta ADC clock sources*
- *Section 34.2.1.5, Sigma-Delta ADC starting/triggering*
- *Section 34.2.1.6, Sigma-Delta ADC external modulator*
- *Section 34.2.1.7, Conversion result interrupt/DMA gating signal*
- *Section 34.2.1.8, Sigma-Delta ADC analog input channel selection*

For detailed module information, refer to the Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface chapter.

34.2.1.1 Sigma-Delta ADC external signal description

The Sigma-Delta ADC external package pins for both the SPC572Lx 100-pin and 80-pin LQFPs are defined in *Table 341*.

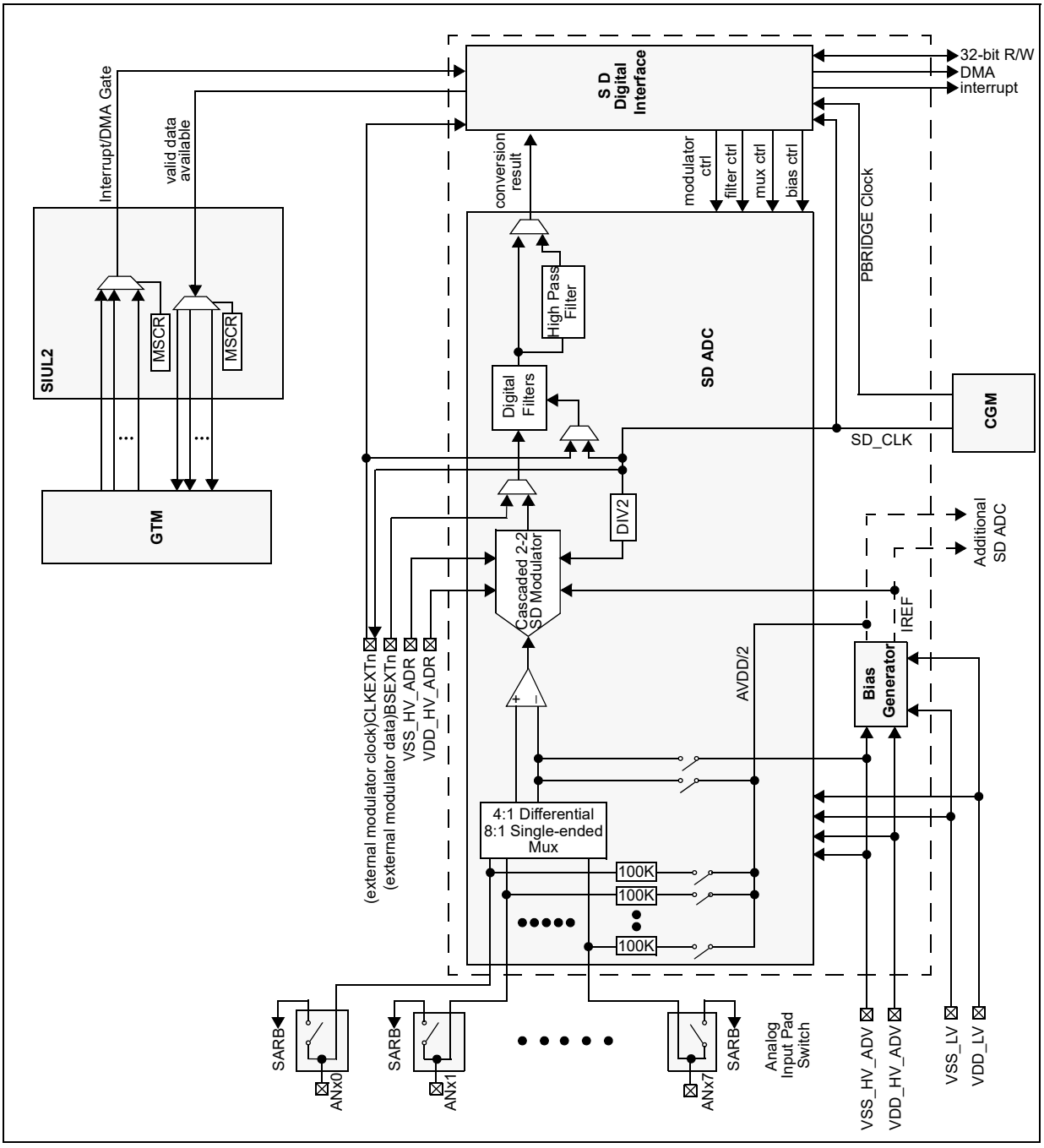
Table 341. Sigma-Delta ADC external signal description

Signal name	Signal type	Signal voltage	Description
AN[n]	Analog input	0–5 V	Single-Ended or Differential Analog Input
VDD_HV_ADV	Analog supply	5 V	Sigma-Delta ADC Analog Power Supply
VSS_HV_ADV	Analog ground	0 V	Sigma-Delta ADC Analog Ground
VDD_HV_ADR	Analog supply	3–5 V	Sigma-Delta ADC Reference
VSS_HV_ADR	Analog ground	0 V	Sigma-Delta ADC Ground Reference
BSEXT[3, 2, 0]	Digital input	0–5 V	Sigma-Delta ADC External Modulator Data Input
CLKEXT[3, 2, 0]	Digital input	0–5 V	Sigma-Delta ADC External Modulator Clock Input

34.2.1.2 Sigma-Delta ADC integration diagram

Figure 304 shows the block diagram of the Sigma-Delta ADC integration on SPC572Lx. Note that in the diagram any unused Sigma-Delta ADC input can be read by SAR ADCB. This is true for the unused input in single-ended operation.

Figure 304. Sigma-Delta ADC integration diagram



34.2.1.3 Sigma-Delta ADC analog input multiplexing

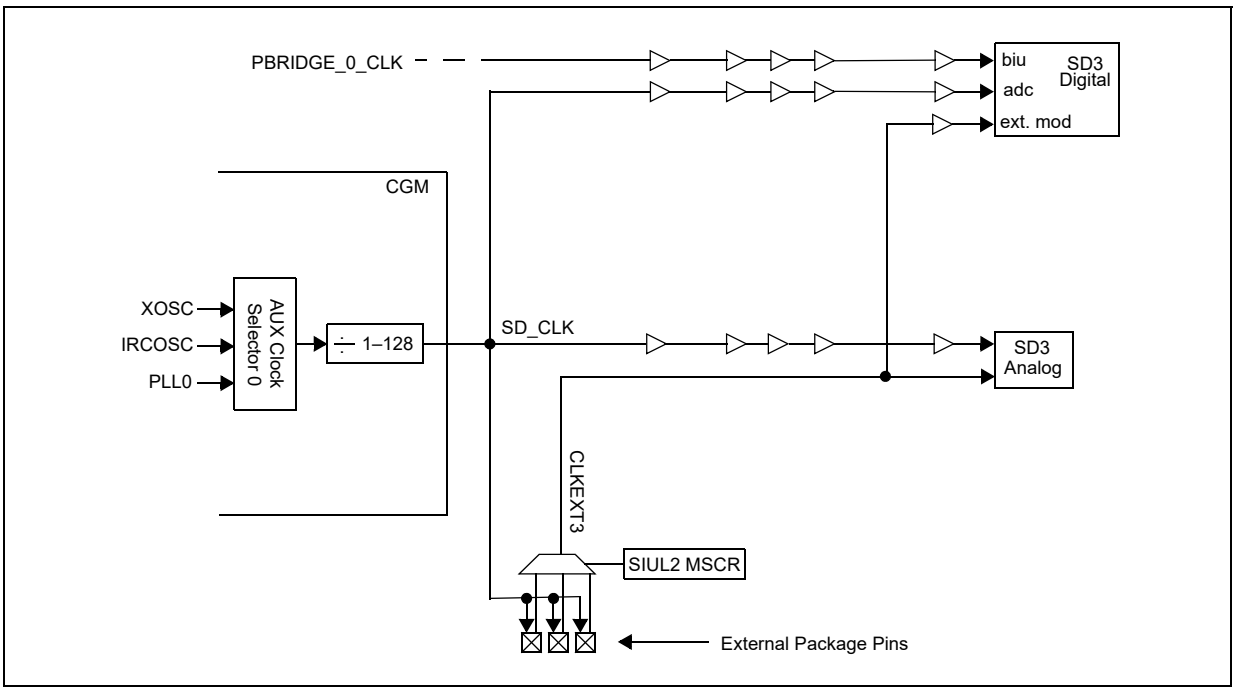
The analog input multiplexing switches are contained within the Sigma-Delta ADC. A single analog switch pad cell is used for the Sigma-Delta analog input pins in order to share the pin with SAR ADCB. The connection to the Sigma-Delta ADC inputs is made to the pad pin through the analog switch pad cell, as shown in *Figure 304*. The mapping of analog input pins to the Sigma-Delta converter is given in *Table 340*.

34.2.1.4 Sigma-Delta ADC clock sources

The Sigma-Delta ADC digital interface on the SPC572Lx has a register clock input, which is separate from the module clock input. The register interface is clocked by the PBRIDGE*n*_CLK. The module clock is sourced by the modulator clock (SD_CLK). See the Clocking chapter for more detail on the clock sources. The input clock for the Sigma-Delta modulator is the SD_CLK. *Figure 305* gives a block diagram of the required Sigma-Delta ADC clock architecture. See *Section 21.6.4: Sigma-Delta ADC clocking*.

The frequency range for the SD_CLK is 4–16 MHz. Both the analog and digital blocks of the ADC contain an internal clock mux for external modulator mode. The external modulator clock (CLKEXT) can be input to the device, or generated from SD_CLK from the CGM. The pad input/output settings in the corresponding SIUL2 MSCR pad control registers determine the external modulator clock source to the ADC blocks. The CLKEXT pins are multiplexed on multiple external pins on the device. Selection of the CLKEXT pin to use for the external modulator clock for each ADC is done in the SIUL2 MSCR input multiplexing registers. The Sigma-Delta ADC has three external pins to select from for an external modulator clock.

Figure 305. Sigma-Delta ADC clock diagram



34.2.1.5 Sigma-Delta ADC starting/triggering

The Sigma-Delta ADC on SPC572Lx supports starting the converter with a software write to an enable bit in a register in the digital interface.

34.2.1.6 Sigma-Delta ADC external modulator

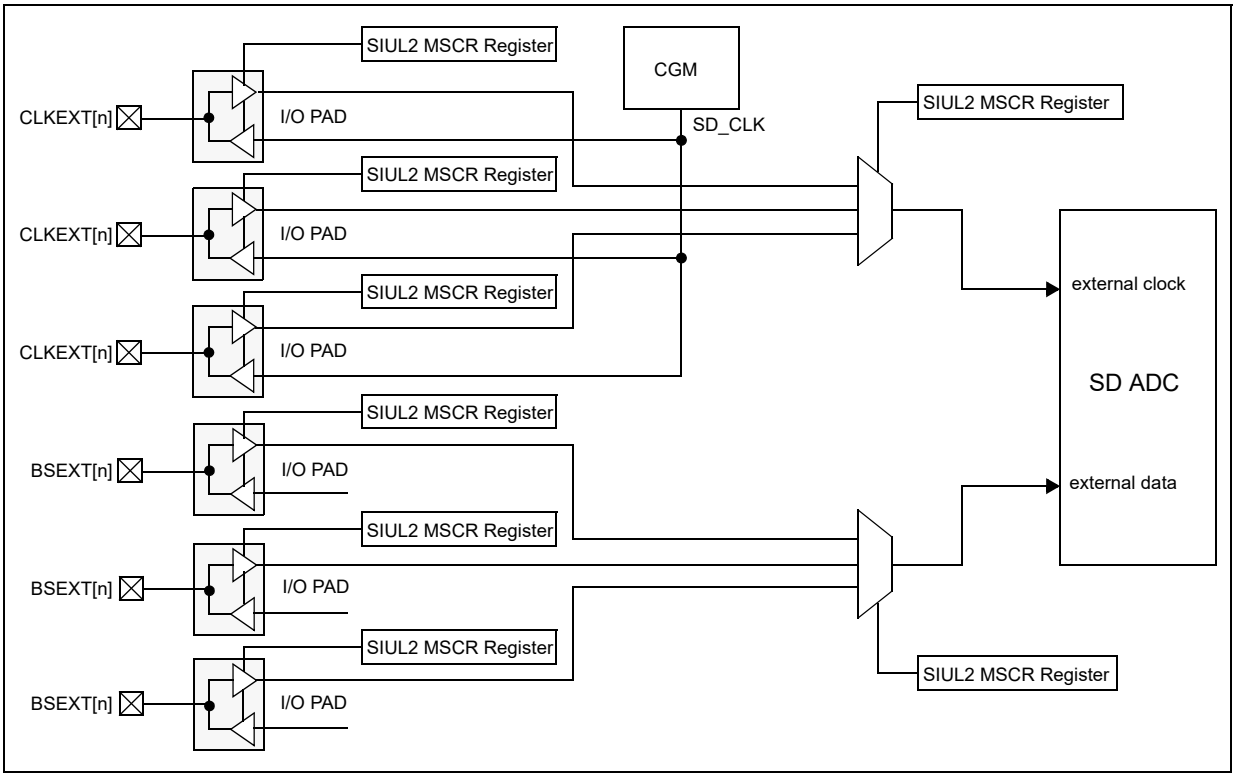
34.2.1.6.1 External modulators

The Sigma-Delta ADC supports an interface to external Sigma-Delta modulators. In this mode, the Sigma-Delta ADC internal modulator is disabled, and a serial bit stream from an external modulator is clocked into the back-end digital filters of the on-chip Sigma-Delta

ADC. Currently, the AMC1203 external modulator is supported by the Sigma-Delta ADC. It is possible that other external second order modulators with clock rates less than 10 MHz and a compatible electrical interface (see SPC572Lx datasheet) could be supported.

The external modulator interface for the Sigma-Delta ADC on SPC572Lx is shown in *Figure 306*.

Figure 306. Sigma-Delta ADC external modulator interface



34.2.1.6.2 Clock/data inputs

Two external pins are required for the Sigma-Delta ADC to support an external modulator, an input data pin BSEXT (bitstream data), and an input/output clock pin CLKEXT (bitstream clock). When the external modulator is used, the clock and data inputs to the Sigma-Delta ADC digital filters come from the CLKEXT and BSEXT pins, instead of the internal modulator, as shown in *Figure 306*. External modulator operation is enabled in the ADC digital interface registers. The internal modulator is disabled. The CLKEXT and BSEXT inputs are multiplexed on port input pins. The Sigma-Delta ADC has three sets of CLKEXT/BSEXT pins, and selection of those pins is done in the SIUL2 MSCRs.

34.2.1.6.3 Clock output

SPC572Lx provides an option for an output clock from the device to the external modulator. When providing an output clock, the SD_CLK from the CGM is connected to the CLKEXT output pin. Both the input and output buffer of the CLKEXT pin must be enabled in the corresponding SIUL2 MSCR, in order for both the external modulator and internal filters to be clocked from the SD_CLK. This is shown in *Figure 306*.

34.2.1.7 Conversion result interrupt/DMA gating signal

The Sigma-Delta ADC digital interface has an input signal that is used to disable interrupt and DMA requests for conversion results. The input signal is connected to outputs from the GTM timer block. The digital interface has a register bit for enabling and disabling the interrupt/DMA gating feature.

When the signal from the GTM is asserted, no interrupt or DMA request is generated for any conversion results. The polarity of the gating signal is such that when the signal is high, interrupt and DMA requests are passed. When the gating signal is low, interrupt and DMA requests are blocked.

The gating signal for each ADC is driven by several GTM channels. See *Chapter 13: System Integration Unit Lite2 (SIUL2)* for the specific mapping of GTM channels to Sigma-Delta interrupt/DMA gating signals.

34.2.1.8 Sigma-Delta ADC analog input channel selection

For the Sigma-Delta mux inputs AN[x], the positive and negative input terminals are selected in the Sigma-Delta ADC digital interface Channel Selection Register (CSR).

This section shows analog channel selections for each combination of Module Configuration Register (MCR) field values MCR_MODE and MCR_VCOMSEL. *Table 342* provides a quick reference key to the MCR fields used in the SD ADC analog input selection tables.

Table 342. Register fields for SD ADC analog input channel selection

Field	Description
MCR_VCOMSEL	Common Voltage Bias Selection This bit selects the common voltage bias for the negative input terminal of SDADC during single-ended mode (MODE=1). 0 Negative input terminal is biased with VREFN. 1 Negative input terminal is biased with VREFP/2 (half scale bias).
MCR_MODE	Mode selection 0 Differential input mode is selected. 1 Single-ended input mode is selected.
CSR_ANCHSEL	Analog Channel Selection Based on single-ended or differential mode of operation (MODE bit of MCR), common mode voltage selection (VCOMSEL bit of MCR), this bit-field defines the connectivity of analog inputs to either positive or negative polarity terminals of Sigma-Delta ADC, as shown in *Table 343*.

Table 343. SD ADC3 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	
0	0/1	000	AN[0]	AN[1]
		001	AN[2]	AN[3]
		010	AN[4]	AN[5]
		011	AN[6]	AN[7]
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

34.2.2 Successive Approximation Register Analog-to-Digital Converter (SARADC)

There are three independent SAR ADCs on SPC572Lx. Refer to *Figure 303* for the ADC subsystem block diagram.

The following sections describe configuration details:

- *Section 34.2.2.1, SAR ADC external signal description*
- *Section 34.2.2.2, SAR ADC integration diagram*
- *Section 34.2.2.3, SAR ADC analog input pin multiplexing*
- *Section 34.2.2.4, SAR ADC clock sources*
- *Section 34.2.2.5, SAR ADC triggering*
- *Section 34.2.2.6, SAR ADC alternate reference*
- *Section 34.2.2.7, SAR absolute voltage reference*
- *Section 34.2.2.8, SAR ADC diagnostic*
- *Section 34.2.2.9, SAR ADC channel assignment*
- *Section 34.2.2.10, SAR ADC register definitions*

For detailed module information, refer to the Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface chapter.

34.2.2.1 SAR ADC external signal description

The external package pins connecting to the SAR ADC modules on the device are defined in *Table 344*.

Table 344. SAR ADC external signal description

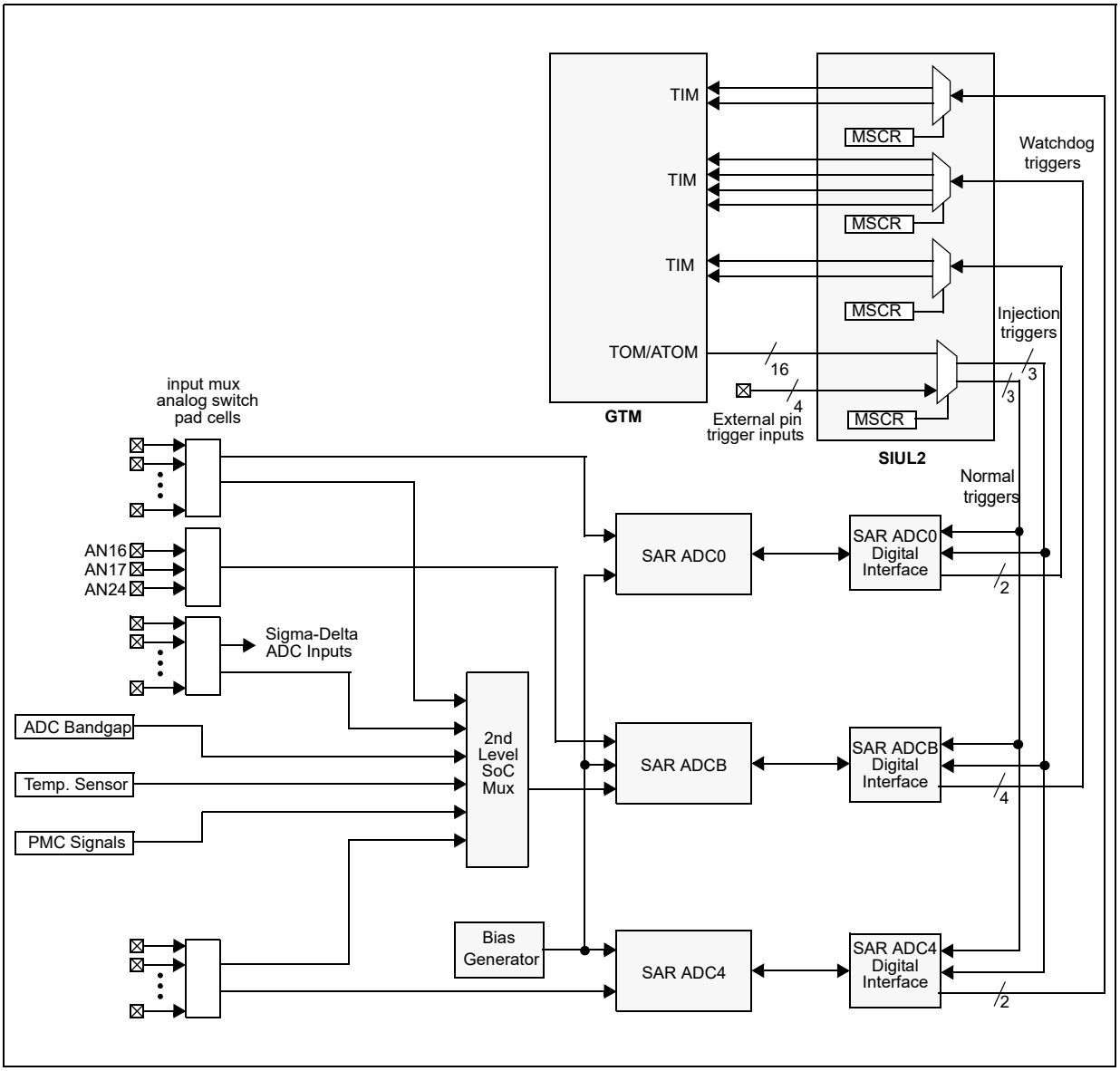
Signal name	Signal type	Signal voltage	Description
AN[n] (see *Table 310*)	Analog input	0–5 V	Single-Ended Analog Input with Pull Up/Down
AN[4:6]	Analog input	0–5 V	Single-Ended Analog Input with no alternate reference selection allowed
AN[7]	Analog input	0–5 V	Single-Ended Analog Input with no alternate reference selection allowed, and switch to ground safety feature
VDD_HV_ADV	Analog supply	5 V	SAR ADC Power Supply
VSS_HV_ADV	Analog ground	0 V	SAR ADC Ground Supply
VDD_HV_ADR	Analog supply	2–5 V	SAR ADC Reference
VSS_HV_ADR	Analog ground	0 V	SAR ADC Ground Reference
DECODE_EXTCH[2:0]	Digital output	0–5 V	SAR ADC0 External Mux Address Output

34.2.2.2 SAR ADC integration diagram

Figure 307 shows the block diagram of the integration of the three SAR ADCs on SPC572Lx.

SAR ADCB implements a Bias Generator converting the 3/3 × ($V_{DD_HV_ADR}$ – $V_{SS_HV_ADR}$), 2/3 × ($V_{DD_HV_ADR}$ – $V_{SS_HV_ADR}$), 1/3 × ($V_{DD_HV_ADR}$ – $V_{SS_HV_ADR}$), and $V_{SS_HV_ADR}$ reference points through the internal 20 kΩ source impedance. See the SAR ADC block diagram in the Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface chapter.

Figure 307. SAR ADC integration diagram



34.2.2.3 SAR ADC analog input pin multiplexing

All analog input pins routed to SAR ADCB and another fast SAR ADC are multiplexed with a dual analog input switch pad cell.

Note: Simultaneous sampling of two ADCs on same analog input is not allowed.

A second level of multiplexing is required for SAR ADCB to reduce the parasitic capacitance on the analog input due to the 48:1 input mux. The second level muxing is done in an SoC level IP block so that it can be optimally placed for routing. The analog switches in the second level mux block are controlled by the SAR ADC digital interface block.

The mapping of analog input pins to the SAR converters is given in *Table 340*.

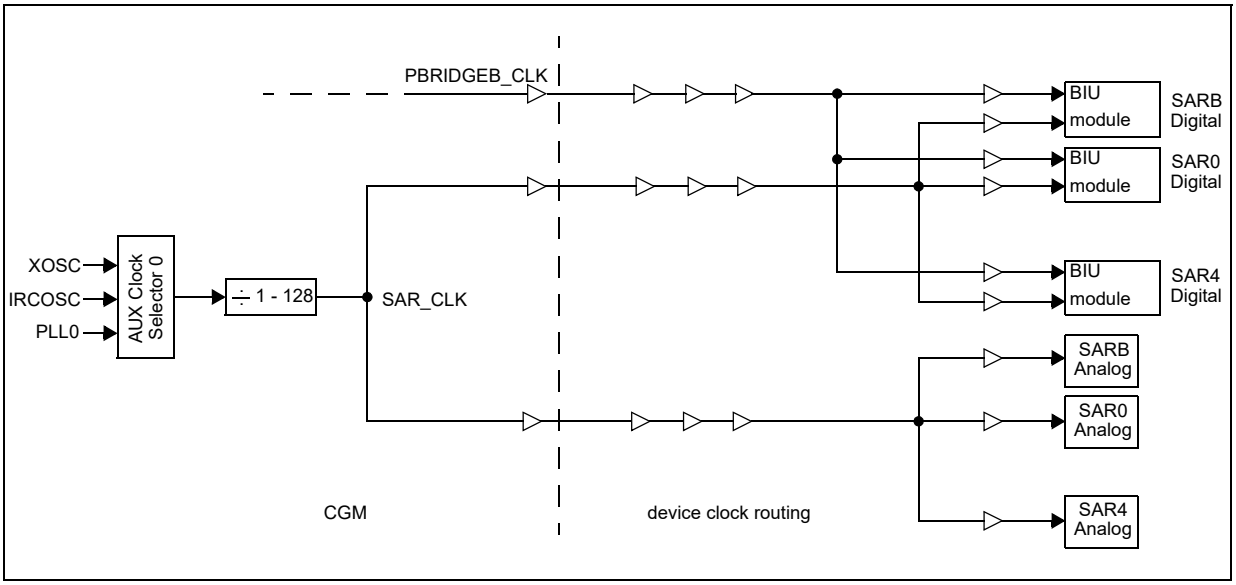
34.2.2.4 SAR ADC clock sources

Each SAR ADC digital interface on SPC572Lx has a register clock input, which is separate from the module clock input. The register interface is clocked by the PBRIDGE*n*_CLK. The module clock is sourced by the ADC clock (SAR_CLK). See *Section 21.2: Clock generation* for more detail on the clock sources.

The input clock for the SAR ADC analog blocks is clocked by the SAR_CLK. *Figure 308* shows a block diagram of the SAR ADC clock architecture. The maximum frequency for the clock input to the converter is defined in the device datasheet.

To avoid crosstalk issues, all SAR ADCs run at the same clock frequency. The SAR digital interface is clocked from the PBRIDGE*n*_CLK clock. All SAR analog blocks are clocked from the output of a single divider of the SAR_CLK at the device level.

Figure 308. SAR ADC clock diagram



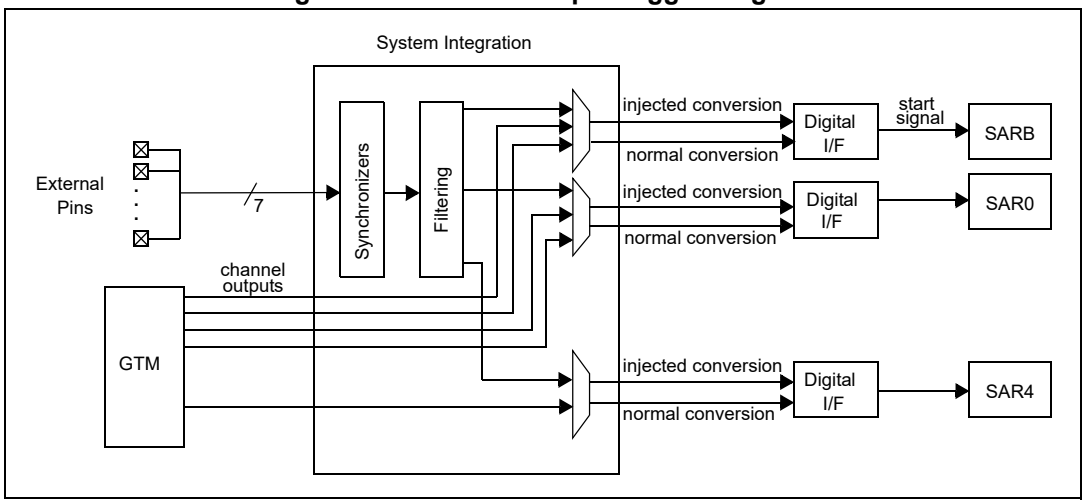
34.2.2.5 SAR ADC triggering

Each SAR ADC 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 SAR ADC on the device has an independent set of trigger inputs, but the same sources can be used by each SAR ADC and the trigger inputs are configurable by SIUL MSCR registers associated to the SAR ADC.

The trigger sources for each SAR ADC on the device are either external input pins or GTM channels. Each external input can selectively trigger any of the three SAR ADCs on the device. The synchronization to the PBRIDGE clock and digital filtering of the external input pin triggers is done in the system integration logic.

Each trigger input supports software selectable options for rising edge, falling edge, rising and falling edge, or level based trigger assertion. The trigger selection for each SAR ADC and edge/level selection is located in the SIUL2. The SPC572Lx SoC diagram for the SAR ADC trigger inputs is given in *Figure 309*.

Figure 309. SAR ADC input trigger diagram



Trigger source selection for each SAR ADC is given in the SIUL2 chapter of this document.

34.2.2.6 SAR ADC alternate reference

The SAR ADCs support connection to more than one conversion reference. The alternate references are muxed with analog input pins on SPC572Lx. The analog input pins used as alternate references to the respective SAR ADCs are given in *Table 345*.

Table 345. SAR ADC alternate references

SAR ADC	Alternate reference pin
SARB	AN[16]

To meet safety requirements, the reference for specific SAR analog inputs cannot be changed. For SPC572Lx, these are SARB channels 4–7 and 35, and all other SAR0 and SAR4 analog inputs. For these SAR ADC instances that do not enable an alternate reference, ICDR*n*[REFSEL] is a read-only bit. There is no option to connect these pins as analog references for the device for safety considerations.

Alternate reference selection is enabled only for the following channels, for which ICDR*n*[REFSEL] is a read-write bit:

SARB channels (0–8, 16–17, 24, 36–39, 52–53, 56–61): see *Table 351: SAR ADCB register definitions*.

34.2.2.7 SAR absolute voltage reference

SPC572Lx provides an internal, absolute voltage reference that can be read by SARB. The ADC conversion channel number for the absolute reference, ADC Bandgap Reference, is given in *Table 347*.

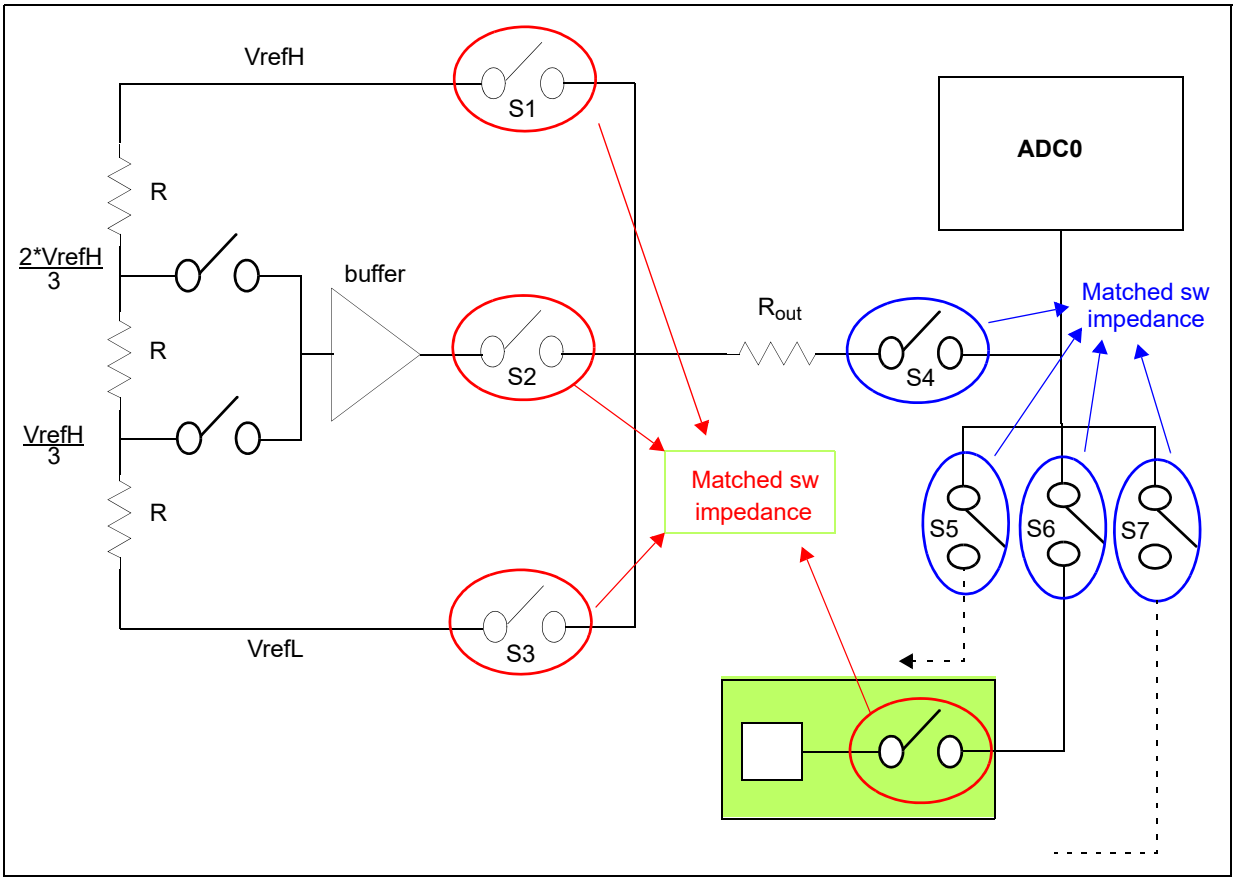
34.2.2.8 SAR ADC diagnostic

34.2.2.8.1 Self Test features

Following are the Self Test features:

- The ADCBIAS provides four different voltage levels defined as GND, $V_{ref}/3$, $2 \times V_{ref}/3$ and V_{ref}. The tolerance of these middle two generated voltages is 4% of V_{ref} (V_{ref} considered to be from 4 V to 6 V).
- The impedance requirement has been satisfied by putting a physical poly resistor R_{out} = 20 kΩ typical, characterized by a process and temperature variation of ±30%.
- The impedance of the switches 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 is matched with the one of the switches that connect the device pin to the ADC input.
- The sampling time of ADC0 needs to be extended to 2.2 ms due to long settling time induced by the high impedance of the source (20 kΩ).
- The SAR ADC Digital Interface provides the necessary control to select the required switches S1–S7 (see *Figure 310: Self Test implementation*) to perform the Self Test.

Figure 310. Self Test implementation



34.2.2.8.2 Internal reference

The SARB ADC provides the ability to sample and convert four internal voltages through a 20 kΩ source impedance. These voltages are as follows:

- $V_{DD_HV_ADR}$
- 1/3 $V_{DD_HV_ADR}$
- 2/3 $V_{DD_HV_ADR}$
- $V_{SS_HV_ADR}$

Conversion of an internal reference voltage works the same way as a normal conversion, with the exception of the conversion time. Due to the 20 kΩ source impedance, the sampling time is much greater than the normal time.

Enabling the sampling and selection of an internal reference is independent of input channel selection for SARB.

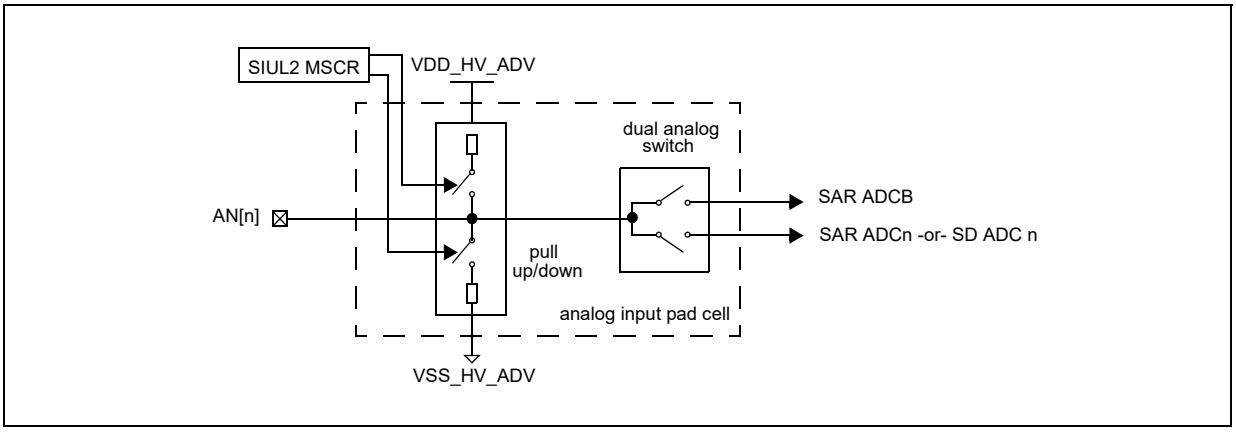
34.2.2.8.3 Analog input pin programmable pull up/down

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. This allows for application testing of the mux input logic. The analog input pin diagram with the pull up/down is shown in *Figure 311*. The pull up/down is included inside the analog input pad cell. The pull up/down is also included in the single switch analog input pad cell.

Control of the pull up/down is done independently in the SIUL2 MSCRs for each analog input pin.

Only the analog input pins specified in *Table 344* have the pull up/down feature.

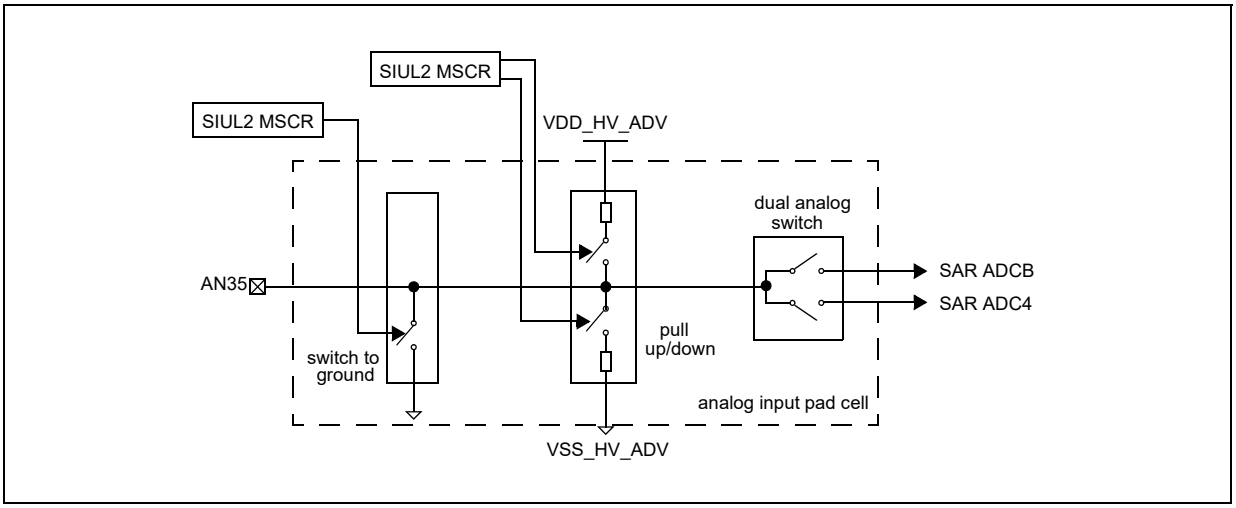
Figure 311. Analog input pin pull up/down pad cell



34.2.2.8.4 Analog input pin switch to $V_{SS_HV_ADV}$

Two analog input pins on SPC572Lx have a programmable switch to $V_{SS_HV_ADV}$ to discharge the input. This switch is implemented on analog input pins AN7 and AN35 for SPC572Lx. The analog input diagram for AN35 is given in *Figure 312*. The default state of the switch for both inputs is OFF.

Figure 312. Analog input AN35 — discharge switch to AVSS



The ground switches for AN7/AN35 are controlled independently from the input switch to the ADC. The ground switch for AN7/AN35 is controlled in input multiplexing MSCR registers in the SIUL2. The MSCR provides selection between open or closed switch, with open being the default state.The ground switches can be used to periodically discharge the associated channels.

34.2.2.9 SAR ADC channel assignment

The input multiplexer channel assignments for the SAR ADCs are provided in *Table 346*.

Table 346. SAR ADC analog input channel assignment

Analog input pin	SARADC_B input channel(1)	SAR ADC module	SAR ADC channel	Description
AN[0:8], AN[16:17], AN[24], AN[36:39], AN[52:53], AN[56:61],	0–8, 16–17, 24, 36–39, 52–53, 56–61	SARADC_B	0–8, 16–17, 24, 36–39, 52–53, 56–61	Analog input pin
AN[0:8]	0–8	SARADC_0	0–8	Analog input pin
AN[52:53], AN[56:61],	52–53, 56–61	SARADC_4	52–53, 56–61	Analog input pin

1. Not all SARADC_B input channels are implemented on all packages—see also *Table 340*.

34.2.2.9.1 Internal voltage monitor channel assignments

Each internal supply monitor (see *Section 9.5: Supply monitoring (POR and LVDs)*) or other test signal has a unique test channel assignment, and is treated by software like any other A/D conversion. Access to the internal test channels is provided on SARB. The internal test channel assignments for SARB are given in *Table 347*.

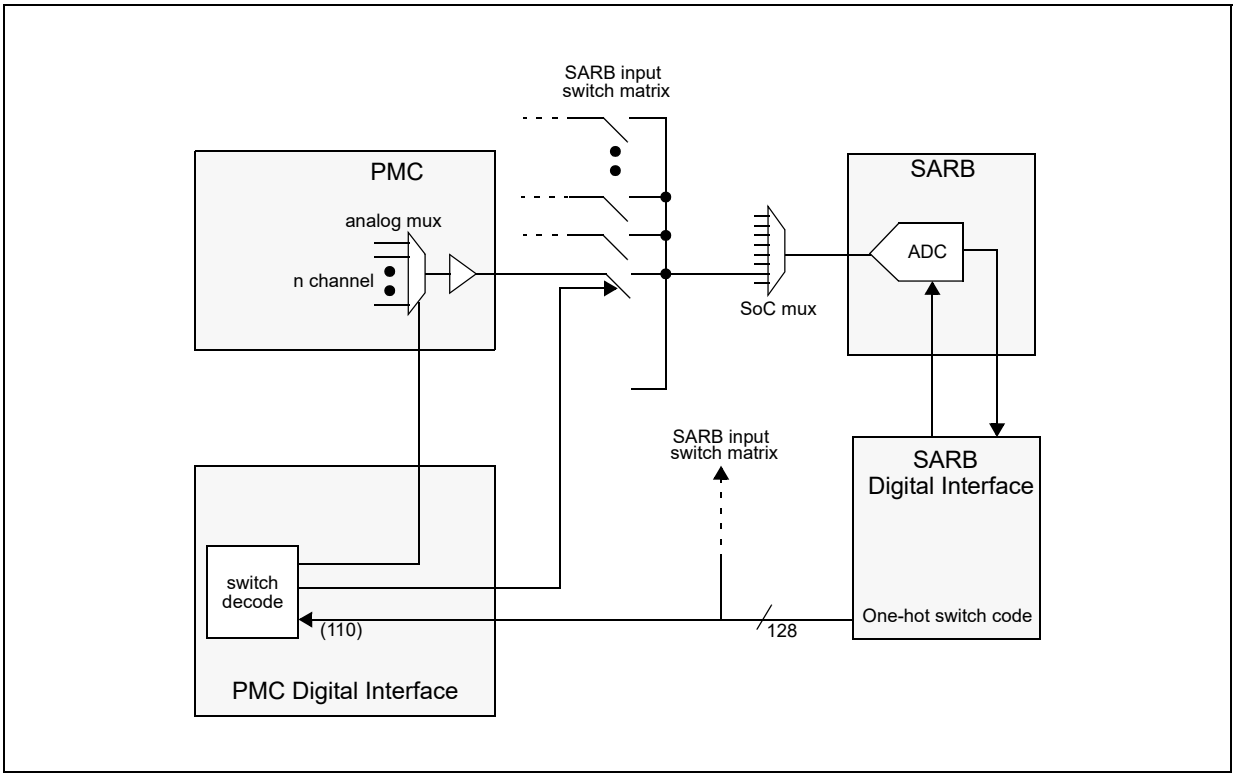
Table 347. SARB analog test channel assignment

SARB input channel	Description
96	VDD_HV_PMC
97–103	Reserved
104	VDD_LV
105–108	Reserved
109	VSS_HV_ADV (ADC Ground Voltage)
110–119	Reserved
120	Temperature Sensor
121	ADC Bandgap Reference
122–123	Reserved
124	SAR BIAS 0—$V_{SS_HV_ADR}$ through 20 kΩ source impedance
125	SAR BIAS 1—1/3 × ($V_{DD_HV_ADR}$ – $V_{SS_HV_ADR}$) through 20 kΩ source impedance
126	SAR BIAS 2—2/3 × ($V_{DD_HV_ADR}$ – $V_{SS_HV_ADR}$) through 20 kΩ source impedance
127	SAR BIAS 3—3/3 × ($V_{DD_HV_ADR}$ – $V_{SS_HV_ADR}$) through 20 kΩ source impedance

34.2.2.9.2 SARB interface to PMC

The connection of the PMC to SARB ADC is shown in *Figure 313*. SARB ADC channel 110 connects to the PMC. The PMC contains an analog mux that passes internal voltage nodes in addition to those given in *Table 347*. The PMC mux is programmed in a register in the PMC digital interface. To convert an internal PMC voltage, the user programs the PMC analog mux selection, and then programs SARB to convert PMC channel 110. See the Power Management chapter of this document for more info on additional internal voltages that can be read by the SARB ADC.

Figure 313. SARB ADC read of internal PMC voltages



34.2.2.10 SAR ADC register definitions

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

- *Section 34.2.2.10.1, SAR ADCB register definitions*
- *Section 34.2.2.10.2, SAR ADC0 register definitions*
- *Section 34.2.2.10.3, SAR ADC4 register definitions*

Table 348 provides a key for the register definition tables. Test channels and external channels are only used for SAR ADCB.

Table 348. SAR ADC 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

Table 348. SAR ADC register descriptions(Continued)

Name	Description	User access
ICDSR*n*	Internal Channel DMA Select Registers	read/write
WTHRHLR0–3	Watchdog Threshold Registers 0–3	read/write
CTR*n*	Conversion Timing Registers 0–3	read/write
ICNCMR*n*	Internal Channel Normal Conversion Mask Registers	read/write
ICJCMR*n*	Internal Channel Injected Conversion Mask Registers	read/write
PDEDR	Power Down Exit Delay Register	read/write
ICDR*n*	Internal Channel Data Registers 0–95	mixed
WTHRHLR4–15	Watchdog Threshold Registers 4–15	read/write
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 349. SAR ADC test channel register descriptions (SARB 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

Table 350. SAR ADC external channel register descriptions (SARB only)

Name	Description	User access
ECDSDR	External Channel Decode Signals Delay Register	read/write
ECIPR*n*	External Channel Interrupt Pending Registers	read
ECIMR*n*	External Channel Interrupt Mask Registers	read/write
ECDSR*n*	External Channel DMA Select Registers	read/write
ECNCMR*n*	External Channel Normal Conversion Mask Registers	read/write
ECJCMR*n*	External Channel Injected Conversion Mask Registers	read/write
ECWSELR*n*	External Channel Watchdog Select Registers	read/write

Table 350. SAR ADC external channel register descriptions (SARB only)(Continued)

Name	Description	User access
ECWENR*n*	External Channel Watchdog Enable Registers	read/write
ECAWORR*n*	External Channel Analog Watchdog Out of Range registers	read
ECMICR*n*	External Channel Mapping to Internal Channel Registers	read/write
ECDR128–191	External Channel Data Registers 128–191	read/write

34.2.2.10.1 SAR ADCB register definitions

Table 351. SAR ADCB register definitions

Name		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
SARADCB_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																
SARADCB_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																
SARADCB_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

Table 351. SAR ADCB register definitions(Continued)

Name		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
SARADCB_ICIPR0	R	0	0	0	0	0	0	0	EOC_CH[24]	0	0	0	0	0	0	EOC_CH[17]	EOC_CH[16]
	W								w1c							w1c	w1c
	R	0	0	0	0	0	0	0	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
SARADCB_ICIPR1	R	0	0	EOC_CH[61]	EOC_CH[60]	EOC_CH[59]	EOC_CH[58]	EOC_CH[57]	EOC_CH[56]	0	0	EOC_CH[53]	EOC_CH[52]	0	0	0	0
	W			w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c				
	R	0	0	0	0	0	0	0	0	EOC_CH[39]	EOC_CH[38]	EOC_CH[37]	EOC_CH[36]	0	0	0	0
	W									w1c	w1c	w1c	w1c				
SARADCB_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																
SARADCB_ICIMR0	R	0	0	0	0	0	0	0	IM_CH[24]	0	0	0	0	0	0	IM_CH[17]	IM_CH[16]
	W																
	R	0	0	0	0	0	0	0	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 351. SAR ADCB register definitions(Continued)

Name		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
SARADCB_ICIMR1	R	0	0	IM_CH[61]	IM_CH[60]	IM_CH[59]	IM_CH[58]	IM_CH[57]	IM_CH[56]	0	0	IM_CH[53]	IM_CH[52]	0	0	0	0
	W									7							
	R	0	0	0	0	0	0	0	0	IM_CH[39]	IM_CH[38]	IM_CH[37]	IM_CH[36]	0	0	0	0
	W																
SARADCB_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	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
SARADCB_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
SARADCB_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																
SARADCB_ICDSR0	R	0	0	0	0	0	0	0	DS_CH[24]	0	0	0	0	0	0	DS_CH[17]	DS_CH[16]
	W																
	R	0	0	0	0	0	0	0	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																
SARADCB_ICDSR1	R	0	0	DS_CH[61]	DS_CH[60]	DS_CH[59]	DS_CH[58]	DS_CH[57]	DS_CH[56]	0	0	DS_CH[53]	DS_CH[52]	0	0	0	0
	W									7							
	R	0	0	0	0	0	0	0	0	DS_CH[39]	DS_CH[38]	DS_CH[37]	DS_CH[36]	0	0	0	0
	W																

Table 351. SAR ADCB register definitions(Continued)

Name		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
SARADCB_ WTHRHLR0– WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADCB_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																
SARADCB_ICNCMR0	R	0	0	0	0	0	0	0	NCE_CH[24]	0	0	0	0	0	0	NCE_CH[17]	NCE_CH[16]
	W																
	R	0	0	0	0	0	0	0	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																
SARADCB_ICNCMR1	R	0	0	NCE_CH[61]	NCE_CH[60]	NCE_CH[59]	NCE_CH[58]	NCE_CH[57]	NCE_CH[56]	0	0	NCE_CH[53]	NCE_CH[52]	0	0	0	0
	W									7							
	R	0	0	0	0	0	0	0	0	NCE_CH[39]	NCE_CH[38]	NCE_CH[37]	NCE_CH[36]	0	0	0	0
	W																
SARADCB_ICJCMR0	R	0	0	0	0	0	0	0	JCE_CH[24]	0	0	0	0	0	0	JCE_CH[17]	JCE_CH[16]
	W																
	R	0	0	0	0	0	0	0	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																
SARADCB_ICJCMR1	R	0	0	JCE_CH[61]	JCE_CH[60]	JCE_CH[59]	JCE_CH[58]	JCE_CH[57]	JCE_CH[56]	0	0	JCE_CH[53]	JCE_CH[52]	0	0	0	0
	W									7							
	R	0	0	0	0	0	0	0	0	JCE_CH[39]	JCE_CH[38]	JCE_CH[37]	JCE_CH[36]	0	0	0	0
	W																

Table 351. SAR ADCB register definitions(Continued)

Name		0 16	1 17	2 18	3 19	27 20	5 21	6 22	7 23	8 24	9 25	10 26	11 27	12 28	13 29	14 30	15 31
SARADCB_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																
SARADCB_ICDR0–3, ICDR8, ICDR10–13, ICDR16–21, ICDR24–29, ICDR32–33, ICDR36–39, ICDR44–53, ICDR56–61	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADCB_ICDR4–7, ICDR35	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADCB_ WTHRHLR4– WTHRHLR7	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADCB_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																
SARADCB_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		
	W																
SARADCB_ICWSELR2	R	0	0	0	0	0	0	0	0	0	WSEL_CH21			0	WSEL_CH20		
	W																
	R	0	WSEL_CH19			0	WSEL_CH18			0	WSEL_CH17			0	WSEL_CH16		
	W																

Table 351. SAR ADCB register definitions(Continued)

Name		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
SARADCB_ICWSELR3	R	0	0	0	0	0	0	0	0	0	WSEL_CH29			0	WSEL_CH28		
	W																
	R	0	WSEL_CH27			0	WSEL_CH26			0	WSEL_CH25			0	WSEL_CH24		
	W																
SARADCB_ICWSELR4	R	0	WSEL_CH39			0	WSEL_CH38			0	WSEL_CH37			0	WSEL_CH36		
	W																
	R	0	WSEL_CH35			0	0	0	0	0	WSEL_CH33			0	WSEL_CH32		
	W																
SARADCB_ICWSELR5	R	0	WSEL_CH47			0	WSEL_CH46			0	WSEL_CH45			0	WSEL_CH44		
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADCB_ICWSELR6	R	0	0	0	0	0	0	0	0	0	WSEL_CH53			0	WSEL_CH52		
	W																
	R	0	WSEL_CH51			0	WSEL_CH50			0	WSEL_CH49			0	WSEL_CH48		
	W																
SARADCB_ICWSELR7	R	0	0	0	0	0	0	0	0	0	WSEL_CH61			0	WSEL_CH60		
	W																
	R	0	WSEL_CH59			0	WSEL_CH58			0	WSEL_CH57			0	WSEL_CH56		
	W																
SARADCB_ICWENR0	R	0	0	0	0	0	0	0	WEN_CH[24]	0	0	0	0	0	0	WEN_CH[17]	WEN_CH[16]
	W																
	R	0	0	0	0	0	0	0	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																
SARADCB_ICWENR1	R	0	0	WEN_CH[61]	WEN_CH[60]	WEN_CH[59]	WEN_CH[58]	WEN_CH[57]	WEN_CH[56]	0	0	WEN_CH[53]	WEN_CH[52]	0	0	0	0
	W									7							
	R	0	0	0	0	0	0	0	0	WEN_CH[39]	WEN_CH[38]	WEN_CH[37]	WEN_CH[36]	0	0	0	0
	W																

Table 351. SAR ADCB register definitions(Continued)

Name		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
SARADCB_ICAWORR0	R	0	0	0	0	0	0	0	AWOR_CH[24]	0	0	0	0	0	0	AWOR_CH[17]	AWOR_CH[16]
	W								w1c							w1c	w1c
	R	0	0	0	0	0	0	0	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
SARADCB_ICAWORR1	R	0	0	AWOR_CH[61]	AWOR_CH[60]	AWOR_CH[59]	AWOR_CH[58]	AWOR_CH[57]	AWOR_CH[56]	0	0	AWOR_CH[53]	AWOR_CH[52]	0	0	0	0
	W			w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c				
	R	0	0	0	0	0	0	0	0	AWOR_CH[39]	AWOR_CH[38]	AWOR_CH[37]	AWOR_CH[36]	0	0	0	0
	W									w1c	w1c	w1c	w1c				

Table 352. SAR ADCB test channel register definitions

Name		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
SARADCB_TCIPR	R	EOC_CH[127]	EOC_CH[126]	EOC_CH[125]	EOC_CH[124]	0	EOC_CH[122]	EOC_CH[121]	EOC_CH[120]	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c		w1c	w1c	w1c								
	R	0	0	EOC_CH[109]	0	0	0	0	EOC_CH[104]	0	0	0	0	0	0	0	EOC_CH[96]
	W								w1c								w1c
SARADCB_TCIMR	R	IM_CH[127]	IM_CH[126]	IM_CH[125]	IM_CH[124]	0	IM_CH[122]	IM_CH[121]	IM_CH[120]	0	0	0	0	0	0	0	0
	W																
	R	0	0	IM_CH[109]	0	0	0	0	IM_CH[104]	0	0	0	0	0	0	0	IM_CH[96]
	W																
SARADCB_TCDSR	R	DS_CH[127]	DS_CH[126]	DS_CH[125]	DS_CH[124]	0	DS_CH[122]	DS_CH[121]	DS_CH[120]	0	0	0	0	0	0	0	0
	W																
	R	0	0	DS_CH[109]	0	0	0	0	DS_CH[104]	0	0	0	0	0	0	0	DS_CH[96]
	W																
SARADCB_TCNCMR	R	NCE_CH[127]	NCE_CH[126]	NCE_CH[125]	NCE_CH[124]	0	NCE_CH[122]	NCE_CH[121]	NCE_CH[120]	0	0	0	0	0	0	0	0
	W																
	R	0	0	NCE_CH[109]	0	0	0	0	NCE_CH[104]	0	0	0	0	0	0	0	NCE_CH[96]
	W																

Table 352. SAR ADCB test channel register definitions(Continued)

Name		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
SARADCB_TCJCMR	R	JCE_CH[127]	JCE_CH[126]	JCE_CH[125]	JCE_CH[124]	0	JCE_CH[122]	JCE_CH[121]	JCE_CH[120]	0	0	0	0	0	0	0	0
	W																
	R	0	0	JCE_CH[109]	0	0	0	0	JCE_CH[104]	0	0	0	0	0	0	0	JCE_CH[96]
	W																
SARADCB_TCWSELR0	R	0	0	0	0	0	0	0	0	0	WSEL_CH101			0	WSEL_CH100		
	W																
	R	0	WSEL_CH99			0	WSEL_CH98			0	WSEL_CH97			0	WSEL_CH96		
	W																
SARADCB_TCWSELR1	R	0	0	0	0	0	WSEL_CH110			0	WSEL_CH109			0	WSEL_CH108		
	W																
	R	0	0	0	0	0	WSEL_CH106			0	0	0	0	0	WSEL_CH104		
	W																
SARADCB_TCWSELR2	R	0	WSEL_CH119			0	WSEL_CH118			0	WSEL_CH117			0	WSEL_CH116		
	W																
	R	0	WSEL_CH115			0	WSEL_CH114			0	WSEL_CH113			0	WSEL_CH112		
	W																
SARADCB_TCWSELR3	R	0	WSEL_CH127			0	WSEL_CH126			0	WSEL_CH125			0	WSEL_CH124		
	W																
	R	0	0	0	0	0	WSEL_CH122			0	WSEL_CH121			0	WSEL_CH120		
	W																
SARADCB_TCWENR	R	WEN_CH[127]	WEN_CH[126]	WEN_CH[125]	WEN_CH[124]	0	WEN_CH[122]	WEN_CH[121]	WEN_CH[120]	0	0	0	0	0	0	0	0
	W																
	R	0	0	WEN_CH[109]	0	0	0	0	WEN_CH[104]	0	0	0	0	0	0	0	WEN_CH[96]
	W																

Table 352. SAR ADCB test channel register definitions(Continued)

Name		0 16	1 17	2 18	3 19	27 20	5 21	6 22	7 23	8 24	9 25	10 26	11 27	12 28	13 29	14 30	15 31
SARADCB_TCAWORR	R	AWOR_CH[127]	AWOR_CH[126]	AWOR_CH[125]	AWOR_CH[124]	0	AWOR_CH[122]	AWOR_CH[121]	AWOR_CH[120]	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c		w1c	w1c	w1c								
	R	0	0	AWOR_CH[109]	0	0	0	0	AWOR_CH[104]	0	0	0	0	0	0	0	AWOR_CH[96]
	W								w1c								w1c
SARADCB_TCCAPR0	R	ESIC_TCH99	ICSEL_TCH99							ESIC_TCH98	ICSEL_TCH98						
	W																
	R	ESIC_TCH97	ICSEL_TCH97							ESIC_TCH96	ICSEL_TCH96						
	W																
SARADCB_TCCAPR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	ESIC_TCH101	ICSEL_TCH101							ESIC_TCH100	ICSEL_TCH100						
	W																
SARADCB_TCCAPR2	R	0	0	0	0	0	0	0	0	ESIC_TCH106	ICSEL_TCH106						
	W																
	R	0	0	0	0	0	0	0	0	ESIC_TCH104	ICSEL_TCH104						
	W																

Table 352. SAR ADCB test channel register definitions(Continued)

Name		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
SARADCB_TCCAPR3	R	0	0	0	0	0	0	0	0	ESIC_TCH110	ICSEL_TCH110						
	W																
	R	ESIC_TCH109	ICSEL_TCH109							ESIC_TCH108	ICSEL_TCH108						
	W																
SARADCB_TCCAPR4	R	ESIC_TCH115	ICSEL_TCH115							ESIC_TCH114	ICSEL_TCH114						
	W																
	R	ESIC_TCH113	ICSEL_TCH113							ESIC_TCH112	ICSEL_TCH112						
	W																
SARADCB_TCCAPR5	R	ESIC_TCH119	ICSEL_TCH119							ESIC_TCH118	ICSEL_TCH118						
	W																
	R	ESIC_TCH117	ICSEL_TCH117							ESIC_TCH116	ICSEL_TCH116						
	W																
SARADCB_TCCAPR6	R	0	0	0	0	0	0	0	0	ESIC_TCH122	ICSEL_TCH122						
	W																
	R	ESIC_TCH121	ICSEL_TCH121							ESIC_TCH120	ICSEL_TCH120						
	W																

Table 352. SAR ADCB test channel register definitions(Continued)

Name		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
SARADCB_TCCAPR7	R	ESIC_TCH127	ICSEL_TCH127							ESIC_TCH126	ICSEL_TCH126						
	W																
	R	ESIC_TCH125	ICSEL_TCH125							ESIC_TCH124	ICSEL_TCH124						
	W																
SARADCB_ TCDR96–101 TCDR104, TCDR106, TCDR108–TCDR110, TCDR112–114, TCDR116–118, TCDR120–127	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

Table 353. SAR ADCB external channel register definitions

Name		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
SARADCB_ECDSDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	DSD											
	W																
SARADCB_ECIPR0	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	EOC_CH[135]	EOC_CH[134]	EOC_CH[133]	EOC_CH[132]	EOC_CH[131]	EOC_CH[130]	EOC_CH[129]	EOC_CH[128]
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 353. SAR ADCB external channel register definitions(Continued)

Name		0 16	1 17	2 18	3 19	27 20	5 21	6 22	7 23	8 24	9 25	10 26	11 27	12 28	13 29	14 30	15 31
SARADCB_ECIMR0	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	IM_CH[135]	IM_CH[134]	IM_CH[133]	IM_CH[132]	IM_CH[131]	IM_CH[130]	IM_CH[129]	IM_CH[128]
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECDSR0	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	DS_CH[135]	DS_CH[134]	DS_CH[133]	DS_CH[132]	DS_CH[131]	DS_CH[130]	DS_CH[129]	DS_CH[128]
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECNCMR0	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	NCE_CH[135]	NCE_CH[134]	NCE_CH[133]	NCE_CH[132]	NCE_CH[131]	NCE_CH[130]	NCE_CH[129]	NCE_CH[128]
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECJCMR0	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	JCE_CH[135]	JCE_CH[134]	JCE_CH[133]	JCE_CH[132]	JCE_CH[131]	JCE_CH[130]	JCE_CH[129]	JCE_CH[128]
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECWSELR0	R	0	WSEL_CH135			0	WSEL_CH134			0	WSEL_CH133			0	WSEL_CH132		
	W																
	R	0	WSEL_CH131			0	WSEL_CH130			0	WSEL_CH129			0	WSEL_CH128		
	W																

Table 353. SAR ADCB external channel register definitions(Continued)

Name		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
SARADCB_ECWENR0	R																
	W																
	R	0	0	0	0	0	0	0	0	WEN_CH[135]	WEN_CH[134]	WEN_CH[133]	WEN_CH[132]	WEN_CH[131]	WEN_CH[130]	WEN_CH[129]	WEN_CH[128]
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECAWORR0	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	AWOR_CH[135]	AWOR_CH[134]	AWOR_CH[133]	AWOR_CH[132]	AWOR_CH[131]	AWOR_CH[130]	AWOR_CH[129]	AWOR_CH[128]
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADCB_ECMICR0	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	ICSEL_ECH128_135						
	W																
SARADCB_ ECDR128–135	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

34.2.2.10.2 SAR ADC0 register definitions

Table 354. SAR ADC0 register definitions

Name		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
SARADC0_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																
SARADC0_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																
SARADC0_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
SARADC0_ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	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
SARADC0_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																

Table 354. SAR ADC0 register definitions(Continued)

Name		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
SARADC0_ICIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	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								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC0_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
SARADC0_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																
SARADC0_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																
SARADC0_ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	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								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC0_ WTHRHLR0– WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																

Table 354. SAR ADC0 register definitions(Continued)

Name		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
SARADC0_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																
SARADC0_ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	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								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC0_ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	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								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC0_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																
SARADC0_ICDR4–8, ICDR10–13	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC0_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																

Table 354. SAR ADC0 register definitions(Continued)

Name		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
SARADC0_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		
SARADC0_ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	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								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
SARADC0_ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	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

34.2.2.10.3 SAR ADC4 register definitions

Table 355. SAR ADC4 register definitions

Name		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
SARADC4_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																
SARADC4_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																

Table 355. SAR ADC4 register definitions(Continued)

Name		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
SARADC4_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
SARADC4_ICIPR1	R	0	0	EOC_CH[61]	EOC_CH[60]	EOC_CH[59]	EOC_CH[58]	EOC_CH[57]	EOC_CH[56]	0	0	EOC_CH[53]	EOC_CH[52]	0	0	0	0
	W			w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c				
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC4_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																
SARADC4_ICIMR1	R	0	0	IM_CH[61]	IM_CH[60]	IM_CH[59]	IM_CH[58]	IM_CH[57]	IM_CH[56]	0	0	IM_CH[53]	IM_CH[52]	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC4_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
SARADC4_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																

Table 355. SAR ADC4 register definitions(Continued)

Name		0 16	1 17	2 18	3 19	27 20	5 21	6 22	7 23	8 24	9 25	10 26	11 27	12 28	13 29	14 30	15 31
SARADC4_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																
SARADC4_ICDSR1	R	0	0	DS_CH[61]	DS_CH[60]	DS_CH[59]	DS_CH[58]	DS_CH[57]	DS_CH[56]	0	0	DS_CH[53]	DS_CH[52]	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC4_ WTHRHLR0– WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
SARADC4_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																
SARADC4_ICNCMR1	R	0	0	NCE_CH[61]	NCE_CH[60]	NCE_CH[59]	NCE_CH[58]	NCE_CH[57]	NCE_CH[56]	0	0	NCE_CH[53]	NCE_CH[52]	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC4_ICJCMR1	R	0	0	JCE_CH[61]	JCE_CH[60]	JCE_CH[59]	JCE_CH[58]	JCE_CH[57]	JCE_CH[56]	0	0	JCE_CH[53]	JCE_CH[52]	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC4_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																

Table 355. SAR ADC4 register definitions(Continued)

Name		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
SARADC4_ICDR32–33, ICDR35, ICDR44–47	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
SARADC4_ICWSELR4	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	WSEL_CH35			0	0	0	0	0	WSEL_CH33			0	WSEL_CH32		
	W																
SARADC4_ICWSELR5	R	0	WSEL_CH47			0	WSEL_CH46			0	WSEL_CH45			0	WSEL_CH44		
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC4_ICWENR1	R	0	0	WEN_CH[61]	WEN_CH[60]	WEN_CH[59]	WEN_CH[58]	WEN_CH[57]	WEN_CH[56]	0	0	WEN_CH[53]	WEN_CH[52]	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
SARADC4_ICAWORR1	R	0	0	AWOR_CH[61]	AWOR_CH[60]	AWOR_CH[59]	AWOR_CH[58]	AWOR_CH[57]	AWOR_CH[56]	0	0	AWOR_CH[53]	AWOR_CH[52]	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

35 Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface

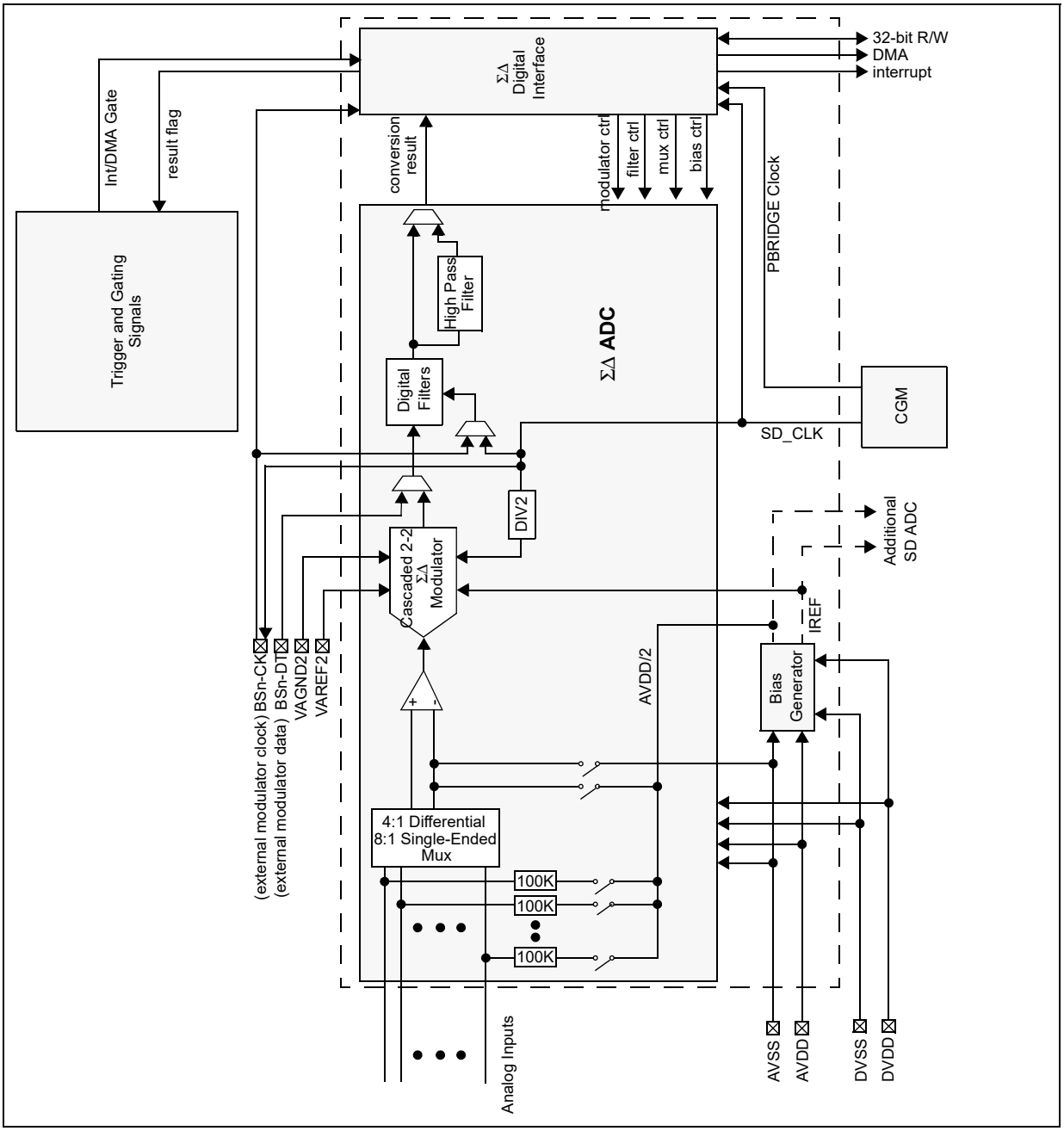
35.1 Introduction

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

35.2 Overview

Figure 314 depicts the block diagram of an SDADC and its digital control block. The SDADC digital interface block provides all control information for operating mode selection (single-ended or differential), analog input gain, decimation rate, high-pass filter enable, etc. It also generates the bias controls for common mode voltage selection, to bias the positive and negative terminals to different voltage levels (for example, half scale bias, reference ground, reference supply) for diagnostics. It generates the analog multiplexer control signals for the selected channels. It also provides external modulator support and keeps the internal modulator in low consumption mode. To assist synchronous operation across multiple ADC blocks, hardware trigger support is implemented to start data conversion.

Figure 314. SDADC block diagram



35.3 Features

- 16-bit data resolution output
- Single-ended input or differential input mode of operation
- Programmable wraparound mechanism for both modes of operation
  - Configurable trigger sources: hardware or software
  - Configurable initial entry and wraparound values for the loop
- Configurable biasing for negative input terminal in single-ended mode
- Gain and offset calibration support using fixed bias for input terminals
- Programmable decimation rate
- Programmable gain for analog inputs
- Optional external modulator support
- Optional high-pass filter for pure AC application
- Hardware trigger support for synchronous operation of multiple SDADC blocks
- Trigger event output generation by software
- Programmable FIFO structure for storing 16-bit converted data
- Interrupt/DMA request generation based on various conditions:
  - Data buffer at or above threshold
  - Data buffer overrun
- Low consumption mode
- Optional conversion process halt mechanism on SoC debug request

35.4 Modes of operation

This section describes the different operation modes of the SDADC. When exiting reset, the mode of operation is determined by the Module Configuration Register (MCR).

35.4.1 Differential input mode

This mode is entered by negating the MCR[MODE] bit. In this mode, a pair of analog inputs are connected to positive and negative terminals of ADC modulator. In order to support gain and offset calibration for diagnostics, it is possible in this mode to select fixed bias voltages to both input terminals.

35.4.2 Single-ended input mode

This mode is entered by asserting the MCR[MODE] bit. In this mode, the negative input terminal is biased with a fixed voltage based on selection information in MCR, and the other analog input is connected to the selected external analog channel.

35.4.3 External modulator mode

This mode is entered by asserting the MCR[EMSEL] bit. In this mode, the data stream and clock outputs from the external modulator available on EMDATA/EMCLK input pins are routed directly to the SDADC block. The internal modulator is bypassed in this mode. Only the digital filters of the SDADC will be used for result calculation. The source for the external modulator clock can be the internal on-chip clock source or the direct clock output coming

from the external modulator. The external pin function control for these pins must be configured in the integration logic on a device, and is not included in the SDADC digital interface.

35.5 External signal description

Table 356 shows the list of signals driven by external pins. At the chip integration level, some of the digital and analog signals might share pins or may not be available external to the chip.

Table 356. Signal properties

Name	Port	Function	I/O	Reset	Pull-up
AN[0:7]	SoC-specific	Single-ended analog inputs or Differential pairs of analog inputs	I	—	—
EMDATA	SoC-specific	External modulator data input	I	0	—
EMCLK	SoC-specific	External modulator clock input	I	0	—
VREFP	SoC-specific	Voltage reference for SDADC	I	—	—
VREFN	SoC-specific	Ground reference for SDADC	I	—	—
AVDD	SoC-specific	Analog voltage supply for SDADC	I	—	—
AVSS	SoC-specific	Analog ground supply for SDADC	I	—	—
DVDD	SoC-specific	Digital voltage supply for SDADC	I	—	—
DVSS	SoC-specific	Digital ground supply for SDADC	I	—	—

35.5.1 Detailed signal descriptions

Table 357 provides a detailed description of the external signals.

Table 357. Detailed signal descriptions

Signal	I/O	Description	
AN[0:7]	I	Each analog input can be used as a single-ended analog input or one of a differential pair of analog inputs which is connected to either positive or negative terminals of the SDADC, depending on the mode of operation and polarity selection.	
		State meaning	Analog input
		Timing	Can be sampled at any time by the ADC.
EMDATA	I	Data stream input provided by external modulator.	
		State meaning	Asserted—Data input is 1 Negated—Data input is 0
		Timing	Assertion—May occur at any time, but data stream is valid only when external modulator is selected by MCR[MODSEL] = 1. Negation—Ignored when external modulator is not selected.

Table 357. Detailed signal descriptions(Continued)

Signal	I/O	Description	
EMCLK	I	Clock input provided by external modulator.	
		State meaning	Asserted—clock high level Negated—clock low level
		Timing	Assertion—May occur at any time, but clock is valid only when external modulator is selected by MCR[MODSEL] = 1. Negation—Ignored when external modulator is not selected.

35.6 Memory map and register descriptions

35.6.1 Memory map

This section provides the memory map for the SDADC block.

Table 358. SDADC Digital interface memory map

Offset or address	Register	Register classification (1)	Location
0x0000	Module Configuration Register (MCR)	Saf-Relv	*on page 687*
0x0004	Channel Selection Register (CSR)	Saf-Relv	*on page 690*
0x0008	Reset Key Register (RKR)	No-Saf	*on page 692*
0x000C	Status Flag Register (SFR)	No-Saf	*on page 693*
0x0010	Request Select and Enable Register (RSER)	Saf-Relv	*on page 695*
0x0014	Output Settling Delay Register (OSDR)	Saf-Relv	*on page 696*
0x0018	FIFO Control Register (FCR)	Saf-Relv	*on page 697*
0x001C	Software Trigger Key Register (STKR)	No-Saf	*on page 698*
0x0020	Converted Data Register (CDR)	No-Saf	*on page 699*

1. Saf-Relv = Safety Relevant, No-Saf = Not Safety Relevant.

35.6.2 Register descriptions

This section provides the block's register bit descriptions in address order.

35.6.2.1 Module Configuration Register (MCR)

The Module Configuration register (MCR) consists of different control bits to select the operating modes and various configuration settings which define the behavior of the SDADC block.

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	PDR					0	PGAN			0	0	EMSEL	HPFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TRIGEDSEL		TRIGEN	TRIGSEL				FRZ	0	0	VCOMSEL	WRMODE	GECEN	MODE	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 315. Module Configuration Register (MCR)

Table 359. MCR field descriptions

Field	Description
0–2	Reserved
3–7 PDR	Programmable Decimation Rate This field selects the over-sampling ratio to be applied to support different passbands with a fixed input sampling clock. The output data rate f_d is equal to f_s / (2×OSR), where f_s is the input sampling clock frequency. When external modulator is selected, the output data rate is independent of this field and is fixed to f_s / 256. 00000 OSR = 24 00001 OSR = 28 00010 OSR = 32 00011 OSR = 36 00100 OSR = 40 00101 OSR = 44 00110 OSR = 48 00111 OSR = 56 01000 OSR = 64 01001 OSR = 72 01010 OSR = 75 01011 OSR = 80 01100 OSR = 88 01101 OSR = 96 01110 OSR = 112 01111 OSR = 128 10000 OSR = 144 10001 OSR = 160 10010 OSR = 176 10011 OSR = 192 10100 OSR = 224 10101 OSR = 256 10110 Reserved .... 11111 Reserved

Table 359. MCR field descriptions(Continued)

Field	Description
8	Reserved
9–11 PGAN	Programmable Gain This field selects the gain to be applied to the analog input stage of the SDADC. The effective analog input becomes the input voltage level multiplied by the gain factor. 000 Gain = 1 001 Gain = 2 010 Gain = 4 011 Gain = 8 100 Reserved 101 Reserved 110 Reserved 111 Gain = 16
12	Reserved
13	Reserved
14 EMSEL	External Modulator Selection 0 External modulator data and clock inputs are ignored 1 External modulator data stream and clock inputs are provided to SDADC
15 HPFEN	High Pass Filter Enable 0 High-pass (DC removal) filter is disabled 1 High-pass (DC removal) filter is enabled
16	Reserved
17–18 TRIGEDSEL	Trigger Edge Selection 00 Falling edge of trigger input is selected 01 Rising edge of trigger input is selected 1x Both edges of trigger input are selected
19 TRIGEN	Trigger Enable This field enables the hardware trigger input to initiate a fresh conversion. Upon receiving a trigger event, SDADC reset input is asserted and deasserted synchronously with respect to the peripheral clock. The reset pulse width is fixed to four peripheral clock cycles. The trigger event is ignored if the SDADC internal modulator is not enabled (MCR[EN] = 0) or the external modulator is selected (MCR[EMSEL] = 1). 0 Trigger input is disabled 1 Trigger input is enabled
20–23 TRIGSEL	Trigger Input Selection This field selects which input will be used for hardware-triggered conversions. 0000 Input trigger 0 is selected 0001 Input trigger 1 is selected ... 1111 Input trigger 15 is selected

Table 359. MCR field descriptions(Continued)

Field	Description
24 FRZ	Freeze This field enables stopping the SDADC conversions at the end of the current channel conversion when the SoC enters debug mode. When the SoC enters debug mode, further conversions by the SDADC analog block will be stopped, and converted data output from the SDADC analog block are ignored. 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. 0 Conversions are not stopped 1 Conversions are stopped
25–26	Reserved
27 VCOMSEL	Common Voltage Bias Selection This field selects the common voltage bias for the negative input terminal of the SDADC during single-ended mode (MODE = 1). 0 Negative input terminal is biased with VREFN 1 Negative input terminal is biased with VREFP/2 (half-scale bias)
28 WRMODE	Wrap-Around Mode This bit selects the wraparound mechanism for conversion of programmed sequence of channels. 0 Wraparound mechanism disabled (in this case the default software control mechanism will be enabled) 1 Wraparound mechanism enabled
29 GECEN	Accurate Gain Error Mode Enable This bit selects the accurate gain error mode, and must be set during gain error calibration as well as during conversions requiring accurate gain error mode. It can be kept '0' if gain error calibration mode is not needed. 0 Gain error calibration mode disabled 1 Gain error calibration mode enabled
30 MODE	Mode selection 0 Differential input mode selected 1 Single-ended input mode selected
31 EN	Enable for SDADC block. Channel and conversion configuration values (PDR,PGAN,ODF,ODA,EMSEL,HPFEN,VCOMSEL,GECEN, MODE, BIASEN, ANCHSEL) should be changed only when EN=‘0’. 0 SDADC internal modulator placed in low consumption mode 1 SDADC internal modulator enabled

35.6.2.2 Channel Selection Register (CSR)

The Channel Selection Register (CSR) selects analog inputs AN[0:7] to be connected to positive and negative terminals of the SDADC and controls the biasing of analog inputs to half-scale bias through 100 kΩ resistance.

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	BIASEN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	ANCHSEL_WRAP			0	0	0	0	0	ANCHSEL		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 316. Channel Selection Register (CSR)

Table 360. CSR field descriptions

Field	Description
0–7	Reserved
8–15 BIASEN[7:0]	Bias enable If BIASEN[i] = 1, analog input AN[i] (could be single-ended input or differential analog input) is connected to half-scale bias through a 100 KΩ resistor.
16–20	Reserved
21–23 ANCHSEL_WRAP	Analog Channel Selection Wraparound Value (ANCHSEL_WRAP) When in wraparound mode, this field indicates the maximum value of the wraparound counter, after which it will wrap around back to initial value 000. This value should be programmed before enabling wraparound mechanism (MCR[WRMODE] = 1).
24–28	Reserved
29–31 ANCHSEL	Analog Channel Selection (ANCHSEL) Based on single-ended or differential mode of operation (MCR[MODE]), common mode voltage selection (MCR[VCOMSEL]), this field defines the connectivity of analog inputs to either positive or negative polarity terminals of the SDADC according to the *Table 361*. Wraparound Mode: – This field also indicates the initial entry value for the first loop of the wraparound sequence from where the wraparound counter will start incrementing when wraparound mode is entered (when wraparound mechanism enabled (MCR[WRMODE] = 1)). – If this field is written any time during wraparound mode (when MCR[WRMODE] = 1), the present value of the wraparound counter will be updated with the value of ANCHSEL — the next trigger will act on the updated channel, and the next increment will be on the updated channel. Here, the value of ANCHSEL chosen should be < ANCHSEL_WRAP; otherwise it will be ignored by hardware. – Wraparound will always be from initial counter value 000.

Table 361. Analog input AN[0:7] selection

MODE	VCOMSEL	ANCHSEL	INP (positive terminal)	INM (negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	
0	0/1	000	AN[0]	AN[1]
		001	AN[2]	AN[3]
		010	AN[4]	AN[5]
		011	AN[6]	AN[7]
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

35.6.2.3 Reset Key Register (RKR)

The Reset Key Register (RKR) is the target for a predefined key write operation used to reset the SDADC to start a fresh conversion.

Offset 0x0008 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESET_KEY															
W																
Reset	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

Figure 317. Reset Key Register (RKR)

Table 362. RKR field descriptions

Field	Description
0–15	Reserved
16–31 RESET_KEY[15:0]	Reset Key This field, when written with 0x5AF0, is used to generate the reset for the SDADC block and to reload the internal counter with the start value programmed in OSDR. Any write access to this register which is different from the predefined key is ignored. Read access will always return the inverted key.

35.6.2.4 Status Flag Register (SFR)

The Status Flag Register (SFR) contains status and flag bits. These bits are set by hardware and reflect the status of the SDADC and indicate the occurrence of events that may generate an interrupt or a DMA request. Software can clear some flags by writing ‘1’ to those bits. Writing a ‘0’ to a flag has no effect.

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	ANCHSEL_CNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	DFEF	0	0	0	0	0	CVDF	DFORF	DFFF
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 318. Status Flag Register (SFR)

Table 363. SFR field descriptions

Field	Description
0–12	Reserved
13–15 ANCHSEL_CNT	Analog Channel Selection Counter(ANCHSEL_CNT) This field reflects the current value of the internal Analog Channel Selection Counter. It indicates the present number of the channel selected through the analog multiplexer.
16–22	Reserved
23 DFEF	Data FIFO Empty Flag This field is asserted when there is no dataword in the FIFO. 0 Data FIFO is not empty 1 Data FIFO is empty
24–28	Reserved
29 CDVF	Converted Data Valid Flag This field gives the actual status of data coming from the SDADC block. The converted data is not valid or becomes corrupted whenever there is a change in configuration settings determined by MCR, channel selection programmed in CSR. Any write access to the registers MCR and CSR will reset this flag. This flag is set automatically after generating SDADC reset and internal timer counts OSD cycles of output clock f_d. 0 Data output from SDADC is not valid 1 Data output from SDADC is valid
30 DFORF	Data FIFO Overrun Flag This field indicates that software or DMA failed to prevent FIFO from overflowing with converted datawords. Software can clear this bit by writing ‘1’ to it after reading the FIFO contents through CDR. This flag remains set if software or DMA failed to clear the condition that caused this flag setting. If DFORF is set, further datawords will not be received into FIFO even if sufficient room exists. 0 No overrun has occurred since the last time the flag was cleared 1 Overrun has occurred or the DFORF has not been cleared since the last overrun occurred
31 DFFF	Data FIFO Full Flag This bit is set when the number of converted datawords in the FIFO is more than the number indicated by FCR[FTHLD]. Software can clear this bit by writing ‘1’ to it. This bit is also cleared by an acknowledgement from the DMA controller when DMA request generation is selected. This bit remains set if software or DMA failed to clear the condition that caused this flag setting. Even if DFFF is set, datawords will continue to be received until an overrun condition occurs. Datawords will not be received if the converted data is not valid (CDVF = 0). 0 The number of datawords in FIFO is less than or equal to the number indicated by FCR[FTHLD] 1 The number of datawords in FIFO is greater than the number indicated by FCR[FTHLD] at some point in time since this flag was last cleared **Note:** Whenever MCR[EN] is cleared in order to stop the SDADC or change the channel configuration, it is necessary to also clear the RSER[DFFDIRE] bit along with it. Similarly if DMA or software is too slow in not reading the FIFO datawords resulting in DFFF condition it is advisable to stop the SDADC by clearing the MCR[EN] followed by clearing of RSER[DFFDIRE] bit. Both these action will ensure safe operation.

35.6.2.5 Request Select and Enable Register (RSER)

The Request Select and Enable Register (RSER) serves two purposes:

- Enables the flag bits in SFR to generate DMA requests or interrupt requests.
- Selects the type of request to generate.

Refer to the field descriptions for the type of requests that are supported.

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	DFFDIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GDIGE	0	0	0	0	0	0	0	0	0	0	0	0	CDVEE	DFORIE	DFFDIRE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 319. Request Select and Enable Register (RSER)

Table 364. RSER field descriptions

Field	Description
0–14	Reserved
15 DFFDIRS	Data FIFO Full DMA/Interrupt Request Select When the SFR[DFFF] and SFR[DFFDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected 1 DMA request is selected
16 GDIGE	Global DMA/Interrupt Gating Enable This field determines whether all module interrupts and DMA requests need to be gated by an external gating signal or not. 0 No impact of external gating signal on module DMA/interrupt requests 1 All module DMA/interrupt requests are qualified only when external gating signal is asserted
17–28	Reserved
29 CDVEE	Conversion Data Valid Event Enable The status of SFR[CDVF] gated with RSER[CDVEE] is passed on as an event on “conversion_flag” output port, which can be used at SoC level for indication of valid data conversion window. Once enabled, the event output will remain asserted as long as valid data conversions are ongoing. 0 Event output disabled 1 Event output assertion/deassertion based on the SFR[CDVF] field

Table 364. RSER field descriptions(Continued)

Field	Description
30 DFORIE	Data FIFO Overrun Interrupt Enable This bit enables the SFR[DFORF] field to generate an interrupt request. The final interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt request is disabled when data FIFO overrun condition occurs 1 Interrupt request is enabled when data FIFO overrun condition occurs
31 DFFDIRE	Data FIFO Full DMA/Interrupt Request Enable This field enables the SFR[DFFF] field to generate a request. The DFFDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GDIGE field. To have safe operation this bit should not be set when MCR[EN]=0. Whenever MCR[EN] is cleared this bit should also be cleared if already set and FCR[FRST] can be set to clear the previous FIFO content. 0 Interrupt/DMA request is disabled when data FIFO full condition occurs 1 Interrupt/DMA request is enabled when data FIFO full condition occurs

35.6.2.6 Output Settling Delay Register (OSDR)

The Output Settling Delay Register (OSDR) provides a delay value to qualify the converted output data coming from the SDADC.

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	0	0	0	0	0	0	0	0	OSD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 320. Output Settling Delay Register (OSDR)

Table 365. OSDR field descriptions

Field	Description
0–23	Reserved
24–31 OSD[7:0]	Output Settling Delay This field defines the delay to qualify the conversion data stored in CDR. Whenever the SDADC block is reset in order to start the conversion from a fresh state, an internal timer is loaded with this start value. The counter counts down with output clock f_d until it reaches zero, and then it generates a flag which qualifies the converted data. The output clock f_d is equal to f_s / (2×OSR), where f_s is the input sampling clock frequency. When external modulator is selected, the output clock is fixed to f_s / 256. **Note:** Refer to the $t_{settling}$ and $t_{LATENCY}$ parameters in the device datasheet for the minimum allowed output settling delay values.

35.6.2.7 FIFO Control Register (FCR)

The FIFO Control Register (FCR) provides the ability to turn on and off the FIFO functionality, and to select the size of FIFO depth. This register also provides the ability to set the programmable threshold for notification of dataword removal from the FIFO. This register may be read at any time, but should only be written when SDADC is disabled (MCR[EN] = 0).

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																FRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	FTHLD				0	0	0	0	FOWEN	FSIZE		FE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 321. FIFO Control Register (FCR)

Table 366. FCR field descriptions

Field	Description
0–14	Reserved
15 FRST	FIFO Flush Reset This is functional soft reset to flush the current FIFO content and initialize the internal FIFO pointers. This is W1S(write one shot) bit which can be written as ‘1’ but always read as ‘0’. Use this option only after disabling the ADC i.e. MCR[EN]=‘0’. 0 : No effect. 1 : Generate a single cycle reset event to flush the FIFO.
16–19	Reserved

Table 366. FCR field descriptions(Continued)

Field	Description
20–23 FTHLD	FIFO Threshold When the number of datawords in the data FIFO is greater than the value in FTHLD field, the FIFO full event is flagged. An interrupt or a DMA request will be generated as determined by DFFDIRE and DFFDIRS fields of RSER. For proper operation, the value in the FTHLD field must be set to be less than or equal to the FIFO size as indicated by FSIZE.
24–27	Reserved
28 FOWEN	FIFO Over Write Enable When set the built-in FIFO structure allows the overwriting of new converted data over older converted data after the FIFO has number of entries equal to the FSIZE. In this case overrun will not be seen. 0 Data FIFO OW option disabled. 1 Data FIFO OW option enabled.
29–30 FSIZE	FIFO Size The maximum number of converted datawords that can be stored in the data FIFO before an overrun occurs. This field is read-only. The reset value of this field depends on the implementation parameter to define the FIFO size. 00 FIFO depth = 1 dataword 01 FIFO depth = 4 datawords 10 FIFO depth = 8 datawords 11 FIFO depth = 16 datawords
31 FE	FIFO Enable When this field is set the built-in FIFO structure is enabled. The size of the FIFO structure is indicated by the FSIZE field. If this field is not set then the FIFO depth is set to one dataword regardless of the value in the FSIZE field. 0 Data FIFO is not enabled for multiple datawords; FIFO depth is one dataword 1 Data FIFO is enabled; FIFO depth is indicated by FSIZE

35.6.2.8 Software Trigger Key Register (STKR)

The Software Trigger Key Register (STKR) allows generating the trigger event output by a software write operation.

Offset 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 W	ST_KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 322. Software Trigger Key Register (STKR)

Table 367. STKR field descriptions

Field	Description
0–15	Reserved
16–31 ST_KEY[15:0]	Software Trigger Key This bitfield, when written with 0xFFFF, is used to generate a trigger event output which can be used to trigger conversions of multiple SDADC blocks synchronously depending on the SoC implementation. Any write access to this register which is different from the predefined key is ignored. Read access will always return 0x0000.

35.6.2.9 Converted Data Register (CDR)

The Converted Data Register (CDR) stores the converted data in the data FIFO.

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	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 323. Converted Data Register (CDR)

The conversion result in the CDATA word is a 16-bit, signed value.

Table 368. CDR field descriptions

Field	Description
0–15	Reserved
16–31 CDATA[15:0]	Converted Data Register The converted datawords can be read from FIFO by reading this register. The data width is 16 bits which has actual 16-bit data (bits 15 down to 0) coming from ADC.

35.7 Functional description

The SDADC has four distinct available modes determined by fields in MCR.

- Differential input mode
- Single-ended input mode
- External modulator mode
- Low consumption mode

35.7.1 Differential input mode

After system reset exit, this is the default mode of operation if SDADC is enabled by asserting MCR[EN]. This mode is entered by negating MCR[MODE]. In differential input mode, a pair of analog inputs among AN[0:7] is connected to the input terminals of the SDADC modulator block. The module supports four differential pairs: AN[0,1], AN[2,3], AN[4,5], and AN[6,7].

The modulator block has two input terminals, positive (INP) and negative (INM). The selection of input pair depends on the ANCHSEL field of CSR. The connectivity of each analog input in a given differential pair, whether it is connected to positive or negative terminal, is determined according to the *Table 361*.

The differential input mode can also be used to perform gain and offset calibration, where the input terminals are biased with fixed voltages coming from the bias block.

35.7.2 Single-ended input mode

Single-ended input mode is entered by asserting MCR[MODE]. In this mode, one of the analog inputs among AN[0:7] is selected by the ANCHSEL field of CSR, and is connected to the positive terminal. The negative input terminal is biased with a fixed internal voltage. There are two internal bias voltages available: reference ground (VREFN) and half-scale bias (VREFP/2). The selection of this common mode voltage is controlled by MCR[VCOMSEL].

35.7.3 External modulator mode

This mode is selected by asserting the MCR[EMSEL] field. When external modulator is selected, the internal modulator can be put in low consumption mode by deasserting the MCR[EN] field. In this mode, the data stream and clock outputs from the external modulator available on EMDATA/EMCLK input pins are multiplexed with internal modulator data output and input sampling clock (f_s). External modulator data must be synchronized with respect to the falling edge of EMCLK. In this mode, hardware trigger input is ignored even if it is enabled by asserting the MCR[TRIGEN] field. All other configuration settings (for example gain, decimation rate, filter, etc.) programmed in MCR have no effect on the conversion.

35.7.4 Low consumption mode

This is the default mode after exiting from reset. This mode is exited by asserting the MCR[EN] field to enable the SDADC block. When data conversion is not required, the SDADC internal modulator can be put in low consumption mode to reduce power consumption.

35.7.5 Analog input multiplexing and bias support

The complete selection and connectivity of analog inputs AN[0:7] is based on the ANCHSEL field of CSR and the MODE and VCOMSEL bits of MCR, according to the *Table 361*. Each analog input can be biased to half scale (AVDD/2) via on-chip 100 kΩ resistance. This allows simple AC coupling to the external analog inputs through a capacitor. This biasing is applicable irrespective of whether analog channel is selected through multiplexer or not.

35.7.6 Programmable gain and decimation rate

All analog inputs can be configured to have a selectable input gain as defined in the PGAN field description in MCR. This means the input signal is sampled and the result is amplified by the factor determined by PGAN before providing the same to modulator.

The SDADC module is provided with a fixed input sampling clock f_s. The input signal is grossly oversampled by the modulator in order to reduce quantization noise. To support different passbands with a fixed sampling clock, a programmable decimation rate is implemented. The process of decimation, to eliminate redundant data at the output while retaining the necessary information, is controlled by the field PDR in MCR.

35.7.7 High-pass filter support

For pure AC applications, it is useful to remove any DC component in the input signal. An optional high-pass filter is implemented which can be enabled by asserting the MCR[HPFEN] field. The –3 dB frequency of the filter is fixed (not programmable) and is available in the device data sheet. This high-pass filter is implemented in the decimation filter logic of the SDADC and is applicable only to internal modulator mode.

35.7.8 Data conversion

Software Control Mechanism: The SDADC block will be in continuous conversion mode as soon as it is enabled. The conversion data is not accurate enough for some time, depending on the latency and output settling time specifications. This is true even when there is any change in the configuration settings provided by MCR, change in the analog channel selection due to mux switching time. To account for these delays, an internal timer is implemented which counts down from a start value determined by the OSD field of OSDR. This timer is reloaded whenever the RESET_KEY of RKR is written with 0x5AF0, which is required to start SDADC conversion from a fresh state with a changed configuration. The timer is stalled on reaching count ‘0’ and asserts the SFR[CDVF] field. Once this flag is asserted, converted data is loaded into data FIFO on every rising edge of the output clock (f_d).

Any write access to the MCR and CSR registers will deassert the CDVF flag bit to indicate that data coming from SDADC is not valid or corrupted. If the CDVF flag is reset, the flags DFFF and DFORF are blocked from becoming asserted, i.e. transitioning from inactive state to active state. If DFFF bit is already set, this means the data FIFO is already full before CVDF deassertion.

The DFF flag bit (which not conditioned by the interrupt or DMA selection) is provided outside which can be routed to timer module at SoC level. Wraparound Control Mechanism: Here a wrap-around logic shall include all analog channels of the input multiplexer from Ain(“000“) to Ain(CSR[ANCHSEL_WRAP]) in both Single ended and Differential ended mode in wraparound sequence triggered by Harare or software source:

- MCR[WRMODE] bit would enable or disable the wraparound mechanism. Clearing the WRMODE bit automatically switches FSM back to Software control mode.
- Every valid trigger advances the wraparound counter to next channel to be converted.
- Wraparound Counter: An internal wraparound counter which points to analog channel to be converted takes the initial entry value for the first loop of execution from CSR[ANCHSEL] while entering wraparound mode. From next loop onwards the counter wraparounds to default initial value “000“. Maximum value of the counter (wraparound value) should be programmed in CSR[ANCHSEL_WRAP] before entering

wraparound mode, definition of ANCHSEL and ANCHSEL_WRAP is provided in *Table 361: Analog input AN[0:7] selection*.

- Upon entering or during wraparound mode user should take care to program ANCHSEL value < ANCHSEL_WRAP value, otherwise hardware would reject the new ANCHSEL value.
- Possible trigger sources:
  – SOURCE1(Software Bit): writing to STKR.ST_KEY at any time during the wraparound mode (irrespective of MCR[TRIGEN or TRIGSEL] setting) generates a trigger pulse, each such trigger initiates a fresh conversion and advances the wraparound counter by one.

Note: Care should be taken while programming the TRIGEN/TRIGSEL of each SDADCDig instance that no unintended simultaneous SW triggering of multiple instances happen.

  – SOURCE2(Hardware Trigger): Here external trigger output (for e.g., mux_advance of GTM) enabled by MCR[TRIGEN or TRIGEDSEL or TRIGSEL] before entering wraparound mode for selective or all SDADC instances provides trigger, each such valid trigger initiates a fresh conversion and advances the wraparound counter by one. One of the hw_trig_in ports can be used to connect the external trigger output.
- Next channel number to be converted can be updated by software even in the middle of wraparound sequence execution, this can be done by rewriting CSR[ANCHSEL]. After CSR[ANCHSEL] is rewritten, the next trigger takes updated value (wraparound counter would increment from the updated value).
- Current wraparound counter value can be read from SFR[ANCHSEL_CNT] bitfield.

FIFO Flush Option: This option allows to clear the FIFO content and re-initialize the FIFO pointers before carrying out next set of conversion after reconfiguring the channel and conversion configuration parameters (Gain, Channel no. e.t.c). This can be done by providing a functional reset by setting FCR[FRST] bit.

This allows to clear the stale unread data corresponding to previous conversion and starting afresh from new channel configuration.

35.7.9 Hardware triggering

Software Control Mechanism: SDADC conversion can be triggered by various hardware events coming from either external pins or internal timers. The field TRIGSEL of MCR selects which hardware event is to be used for triggering conversions. This trigger event, which is synchronous to the peripheral clock, is enabled by asserting MCR[TRIGEN]. As soon as a trigger event is detected, reset input of SDADC block is asserted synchronously for four cycles. This will ensure that SDADC starts from a fresh state when reacting to the input trigger event. The SDADC internal modulator must be enabled before enabling the trigger event — otherwise the trigger event will be ignored. A hardware trigger event is also ignored when external modulator is selected (MCR[EMSEL] = 1).

If the same trigger event is provided to multiple SDADC digital interface modules and the respective enable controls (TRIGEN bit MCR) are asserted, synchronous operation of multiple SDADC blocks can be achieved with output results being updated at the same time. Once the reset input is deasserted, the first valid conversion output is stored in data FIFO after a configurable delay determined by OSDR[OSD].

It is also possible to generate a software trigger event output by a write access of STKR with a predefined key. This event output can be used as a trigger to achieve synchronous software start of multiple SDADC blocks.

In case of a hardware trigger, the skew/jitter will be less than 100 ns. It will be more than one ADC clock cycle due to synchronization. In case of a software trigger, all SDADCs start synchronous sampling.

Wraparound Control Mechanism: In wraparound control mode, hardware and software triggers are equally valid. The hardware trigger source should be selected before enabling the wraparound mode.

35.7.10 Interrupt/DMA request support

The SDADC has one condition that generates interrupts only and another that generates interrupts or DMA request.

The DFFF field is asserted when the data FIFO full condition is detected, indicating that the number of converted datawords stored in data FIFO is equal to or greater than the value indicated by threshold determined by FCR[FTHLD]. The interrupt or DMA request is generated only when RSER[DFFDIRE] is asserted. RSER[DFFDIRS] selects whether a DMA request or an interrupt request is generated.

The data FIFO overrun flag (DFORF) indicates the FIFO overflow condition. Further converted datawords cannot be received and data is lost. The interrupt request is generated only when RSER[DFORIE] is asserted.

If RSER[GDIGE] is set, all module interrupt/DMA requests are qualified only when an external global gating signal is asserted.

35.7.11 Gain calibration support

To perform gain calibration, the following sequence needs to be applied.

1. Select differential mode of operation by writing MCR[MODE] to ‘0’.
2. Enable gain error calibration mode by writing MCR[GECEN] to ‘1’.
3. Configure the mux selection ANCHSEL field of CSR to ‘110’. This configuration will apply VREFP on positive terminal and VREFN on negative terminal.
4. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 0x00.
5. Disable the high-pass filter by deasserting MCR[HPFEN].
6. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
7. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. Expected output if there is no gain error is 0b0111_1001_1001_1001, corresponding to full positive scale after attenuation inserted by the internal filter (1*0.95). The measurement should be repeated to reduce contribution of noise during calibration process. D_p is the average value of attenuated positive full scale given by D_p = AVERAGE(CDR[CDATA]).
8. Change the mux selection ANCHSEL field of CSR to ‘111’. This configuration will apply VREFN on positive terminal and VREFP on negative terminal.
9. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
10. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. Expected output if there is no gain error is 0b1000_0110_0110_0110, corresponding to full negative scale after attenuation inserted by the internal filter (-1*0.95). The

measurement should be repeated to reduce contribution of noise during calibration process. Dn is the average value of attenuated negative full scale given by Dn = AVERAGE(CDR[CDATA]).

11. The SDADC calibrated gain can be calculated as:

    Gain = $(D_p – D_n) / 2^{16}$

12. The measured gain value can be used to nullify the gain errors in the digital output. For calibrated conversion, the data CDR[CDATA] provided by the SDADC should be normalized using the calculated gain: $CDATA_{norm}$ is given by $CDATA_{norm}$ = CDR[CDATA]/Gain.

Note: The gain error calibration should only be done with gain equal to '1' (in other words, PGAN = 000). Nevertheless, the calculated gain can be applied to normalized data also for conversion with GAIN>1.

Note: MCR[GECEN] = 1 ensures the accurate gain error mode is enabled. This bit should be set both during calibration process and after calibration, for application data conversion. Application data conversion must then be normalized using the measured gain.

Note: The higher the number of full-scale conversion (D_p, D_n) done, the higher the rejection of noise during the calibration. The number of conversion is dependent on application noise. It is recommended to run at least 16 conversions when calculating an average value.

35.7.12 Offset calibration support

To perform offset calibration, the following sequence must be applied.

1. Select differential mode of operation by writing MCR[MODE] to '0'.
2. Configure the mux selection ANCHSEL field of CSR to '100' or '101' as required.
   a) CSR = '100' in case of data conversion after calibration in "single ended mode with negative input = $V_{SS_HV_ADR_D}$"
   b) CSR = '101' in case of data conversion after calibration in "differential mode" and "single ended mode with negative input = $(V_{DD_HV_ADR_D}–V_{SS_HV_ADR_D})/2$"
3. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 0x00.
4. Disable the high pass filter by deasserting MCR[HPFEN].
5. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
6. Read the digital output stored in FIFO after the output settling time.
7. The measured offset can be used to nullify the offset error in the digital output. Expected output is 0b00_0000_0000_0000. The SDADC offset can be calculated as:

   Offset = Expected Output – Actual Output

Note: The offset must be calculated for each PGAN field setting since it is expected to vary with gain configuration of SDADC.

35.8 Initialization information

To initialize the SDADC registers for data conversion, the following sequence is required.

1. Enable the SDADC by asserting MCR[EN].
2. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
3. Enable high-pass filter if required.
4. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
5. Configure OSD delay according to the SDADC startup time or latency from reset exit.
6. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR. Otherwise, if data conversion need to be triggered by a hardware event, assert MCR[TRIGEN].
7. DFFF interrupt or DMA request will be generated after the data FIFO has reached threshold to indicate that data can be transferred.

The following sequence needs to be followed when external modulator mode of SDADC is selected for data conversions.

1. Disable the SDADC internal modulator by deasserting MCR[EN].
2. Configure MCR to select the external modulator mode.
3. Select the required external pins EMDATA/EMCLK in the system integration logic (SIUL) of the device.
4. Configure OSD delay according to the external modulator startup time or latency from reset exit.
5. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
6. DFFF interrupt or DMA request will be generated after the data FIFO has reached threshold to indicate that data can be transferred.

35.8.1 Data conversion step

To acquire a data from SDADC, the following sequence is required:

1. Enable the SDADC by asserting MCR[EN].
2. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
3. Enable high-pass filter is required.
4. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
5. Configure OSD delay according to the SDADC required startup time or latency from reset exit.
6. Generate a reset event writing 0x5AF0 in RESET_KEY field of RKR register.
7. Wait till FIFO empty flag DFEF of SFR register is clear.
8. Read data by CDATA of CDR register.

Repeat steps from 6 to 8 for new acquisitions.

Note: The time elapsed between reset event and Read data by CDATA field of CDR register are:

- *Reset event <- -> Data Valid flag (CDVS) = Output settling Time delay (Refer to the $t_{settling}$ and $t_{LATENCY}$ parameters in the device datasheet for the minimum allowed output settling delay values.)*
- *Data Valid flag <- -> Data FIFO is not empty = 0.5*CLK_Out + 3*CLK_In*
- *Data FIFO is not empty <- -> read Data (safety point) = 2*CLK_In*

*where:CLK_Out = CLK_In/(2*OSR)*

36 Successive Approximation Register Analog-to-Digital Converter (SARADC)

36.1 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.

36.2 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.

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.
- External channels, mapped to index ranging from 128 to 255.

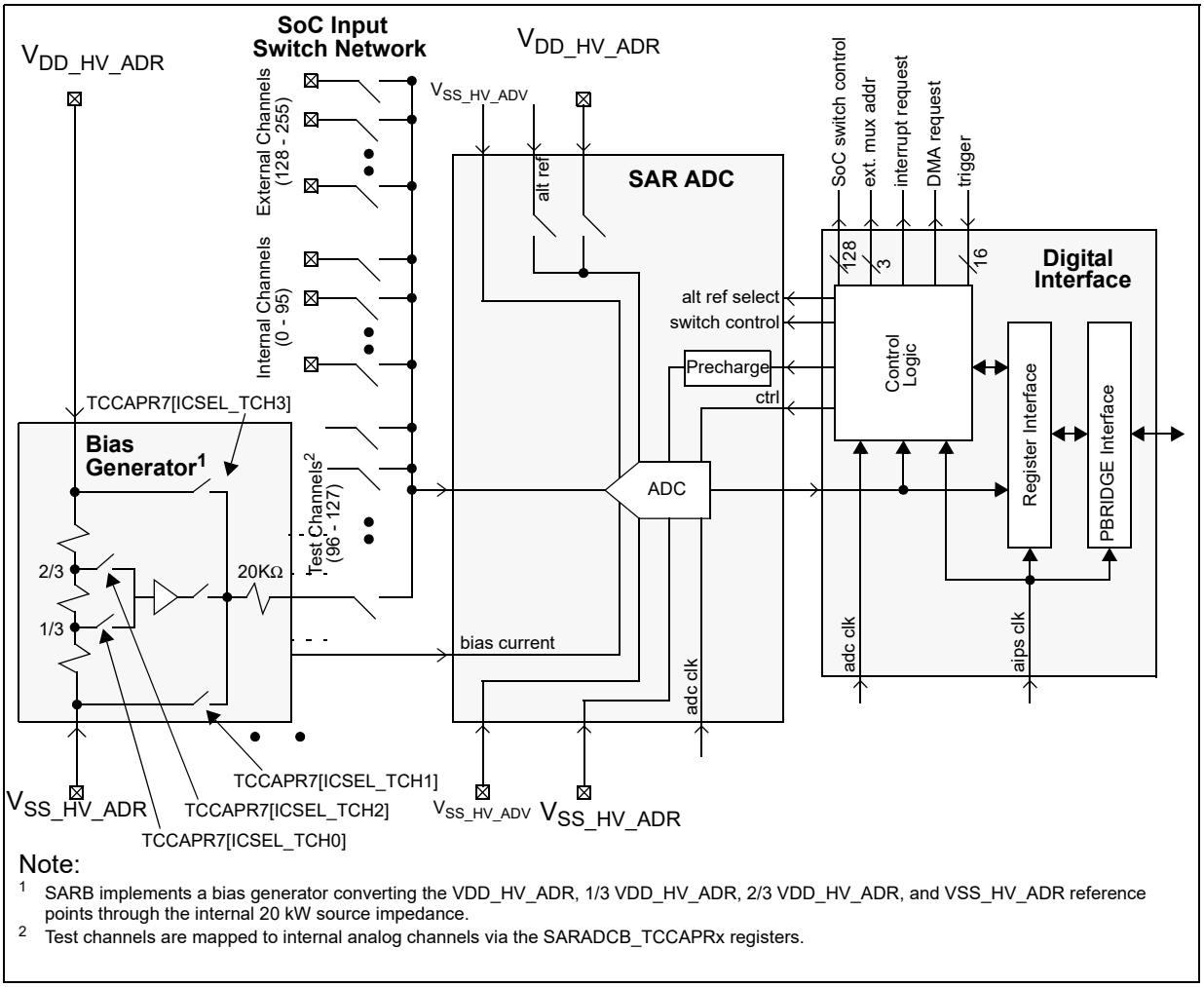
Note: The number of input channels and their mapping depends on the device configuration. See the device-specific Analog-to-Digital Converters (ADC) Configuration section (analog input pin multiplexing and SAR ADC channel assignment) for a channel mapping table.

The mask registers present within the digital interface can be properly programmed to configure which channel has to be converted. External channel selection is provided through external decode signals and are available as alternate functions on GPIO. Each set of eight external channels can be mapped to any internal analog channel via static software programming in specific registers.

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 324 shows the SARADC block diagram.

Figure 324. SARADC block diagram



Note:

1 SARB implements a bias generator converting the VDD_HV_ADR, 1/3 VDD_HV_ADR, 2/3 VDD_HV_ADR, and VSS_HV_ADR reference points through the internal 20 kW source impedance.

2 Test channels are mapped to internal analog channels via the SARADCB_TCCAPRx registers.

36.3 Feature description

36.3.1 Main features

- Selectable 10-bit or 12-bit data resolution output
- Up to 96 internal channels, 32 test channels, 128 external channels supported; 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
- External decode signals (3 signals) for selection of external analog mux inputs
- 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), data valid, data overwritten status
  - Control byte for reference selection, conversion timing parameter selection
- Configurable number of analog watchdogs.
- 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

36.4 Functional description

Two main conversion types are available within the SARADC digital interface:

- Normal conversion
- Injected conversion

36.4.1 Normal channel conversion

This is the normal conversion that the user programs by configuring the normal conversion mask registers (ICNCMR0–2, TCNCMR, ECNCMR0–3). 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).

36.4.1.1 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 one-shot 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.

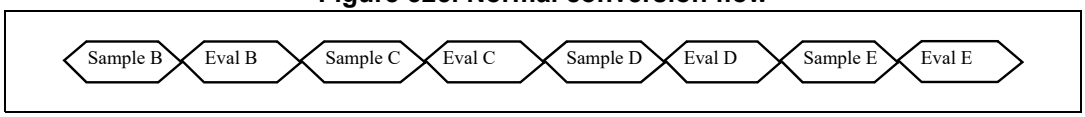
36.4.1.2 Normal conversion operating modes

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 325*.

Figure 325. 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 one-shot 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.

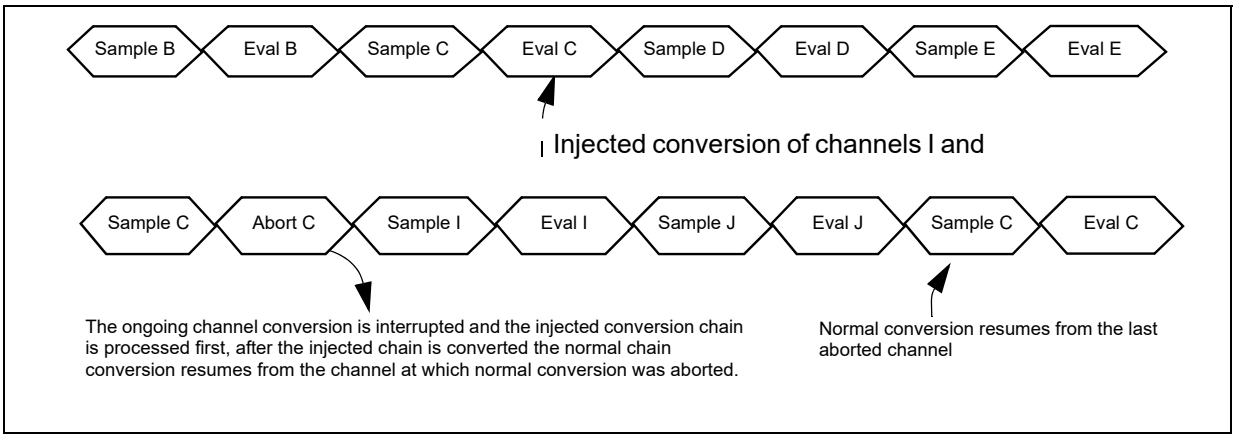
For Example, channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the *Scan Mode*. MODE = 1 is set for Scan Mode. Conversion starts from the channel B followed by conversion of the channels D-E. At the end of conversion of channel E the scanning of channel B 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.

36.4.2 Injected channel conversion

A conversion chain can be injected into the current normal conversion by configuring the injected mask registers (ICJCMR0–2, TCJCMR, ECJCMR0–3). As in normal conversion, each internal, test or external 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 326*.

Figure 326. 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).

36.4.3 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).

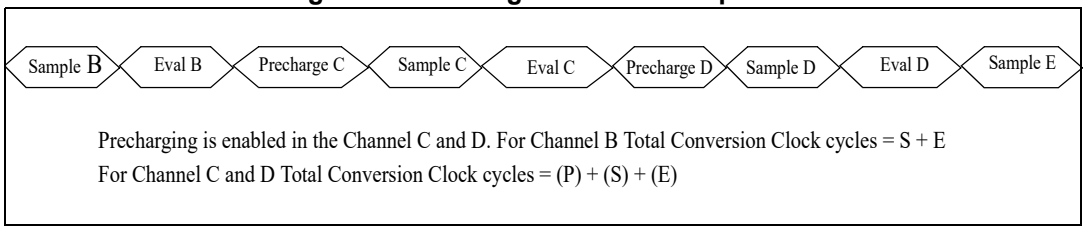
36.4.4 Analog conversion timings and reference selection

36.4.4.1 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 327*. Refer to device datasheet for the minimum precharge and sample durations required.

Figure 327. 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.

36.4.4.2 Alternate reference selection

In order to support different accuracy requirements, a voltage reference can be selected for each channel through REFSEL bit of the corresponding channel data register (for example, ICDR*n*). For some channels, this bit REFSEL is not software programmable and the selection configuration is fixed based on device-specific requirements.

36.4.5 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 the test 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.

36.4.6 External channel mapping to internal analog channel

External channels are supposed to be used in conjunction with a given internal analog channel along with an 8:1 external mux outside the device. Each input of the external mux can be mapped to a unique channel number, thus allowing it to perform conversions like any other channel of the ADC. External channels are mapped from 128 to 255.

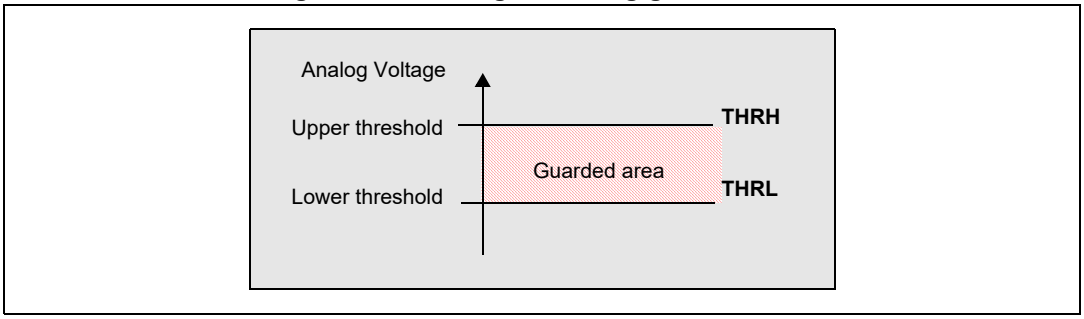
External channels are implemented in the multiples of 8. Each slice of eight channels can be associated to any physical internal analog channel mapped from 0 to 95. This external-to-internal channel mapping is determined by the ICSEL_ECx_x bitfield of ECMICR0–3 registers. For example, when external multiplexer is planned to be used on internal channel 5, and the external inputs need to be mapped to channels 128 to 135, then this channel number 5 must be written to the ICSEL_ECH128_135 bitfield of the ECMICR0 register. After reset, all external channels are mapped to internal analog channel 0. If a conversion on a single external channel (out of group of 8) mapped to a particular internal channel has to be performed than only that particular channel should be enabled in the Conversion Mask Register (ECNCMR or ECJCMR).

After the conversion is performed on a external channel, the converted data is stored at two CDR locations at the end of conversion. The first location is the CDR belonging to the external channel on which conversion is performed and the second location is the CDR belonging to the internal channel on which the external channel is mapped.

36.4.7 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 328*) bounded by upper and a lower threshold values named THRH and THRL respectively.

Figure 328. 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 369*. Depending on the MSKWDGxL and MSKWDGxH mask bits in the WTIMR, an interrupt is generated on threshold violation.

Table 369. 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.

36.4.8 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.

36.4.9 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)

The EOC_CH[x] bit of interrupt pending registers (ICIPR0–2, TCIPR, ECIPR0–3) 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, ECIMR0–3).

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

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.

Conversion should not be initiated unless the required conversion mask bits (ECNCMR or ECJCMR) are set.

36.4.10 External decode signals selection and delay

The ADC provides three external decode signals used as select lines for 8:1 external mux. For each set of 8 external channels (for example [CH128:CH135], [CH136:CH143], and so on), the same 3-bit decode signals output are used to select the inputs of external mux. These external mux select lines change in grey code sequence for each increment of channel selection. This ensures that only one line changes and avoids any current spikes or leakage if more than one line changes simultaneously.

In order to take into account the control switching time of the external analog mux, a Decode Signals Delay register is provided. Writing that register allows programming of the delay between the decoding signal selection and the actual start of conversion.

36.4.11 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.

36.4.12 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.

36.5 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 370. SARADC digital interface register map

Offset	Register	Register classification (1)	Location
0x000	Main Configuration Register (MCR)	Saf-Relv	*36.5.1.1 on page 720*
0x004	Main Status Register (MSR)	No-Saf	*36.5.1.2 on page 723*
0x008–0x00C	Reserved		
0x010	Interrupt Status Register (ISR)	No-Saf	*36.5.1.3 on page 724*
0x014–0x01C	Internal channel Interrupt Pending Registers 0–2 (ICIPR0–ICIPR2)	Saf-Relv	*36.5.1.4 on page 725*
0x020	Interrupt Mask Register (IMR)	Saf-Relv	*36.5.1.5 on page 726*

Table 370. SARADC digital interface register map(Continued)

Offset	Register	Register classification (1)	Location
0x024–0x02C	Internal Channel Interrupt Mask Registers 0–2 (ICIMR0–ICIMR2)	Saf-Relv	*36.5.1.6 on page 727*
0x030	Watchdog Threshold Interrupt Status Register (WTISR)	Saf-Relv	*36.5.1.7 on page 728*
0x034	Watchdog Threshold Interrupt Mask Register (WTIMR)	Saf-Relv	*36.5.1.8 on page 729*
0x038–0x03C	Reserved		
0x040	DMA Enable Register (DMAE)	Saf-Relv	*36.5.1.9 on page 729*
0x044–0x04C	Internal Channel DMA Select Registers 0–2 (ICDSR0–ICDSR2)	Saf-Relv	*36.5.1.10 on page 730*
0x050–0x05C	Reserved		
0x060–0x06C	Watchdog Threshold Registers 0–3 (WTHRHLR0–WTHRHLR3)	Saf-Relv	*36.5.1.11 on page 731*
0x070–0x090	Reserved		
0x094–0x0A0	Conversion Timing Registers 0–3 (CTR0–3)	Saf-Relv	*36.5.1.12 on page 732*
0x0A4–0x0AC	Internal Channel Normal Conversion Mask Registers 0–2 (ICNCMR0–ICNCMR2)	Saf-Relv	*36.5.1.13 on page 732*
0x0B0	Reserved		
0x0B4–0x0BC	Internal Channel Injected Conversion Mask Registers 0–2 (ICJCMR0–ICJCMR2)	Saf-Relv	*36.5.1.14 on page 733*
0x0C0–0x0C4	Reserved		
0x0C8	Power Down Exit Delay Register (PDEDR)	Saf-Relv	*36.5.1.15 on page 735*
0x0CC–0x0FC	Reserved		
0x100–0x27C	Internal Channel Data Registers 0–95 (ICDR0–ICDR95)	Saf-Relv	*36.5.1.16 on page 735*
0x280–0x2AC	Watchdog Threshold Registers 4-15 (WTHRHLR4-WTHRHLR15)	Saf-Relv	*36.5.1.17 on page 737*
0x280–0x2AC	Reserved		
0x2B0–0x2DC	Internal Channel Watchdog Selection Registers 0–11 (ICWSELR0–ICWSELR11)	Saf-Relv	*36.5.1.18 on page 738*
0x2E0–0x2E8	Internal Channel Watchdog Enable Registers 0–2 (ICWENR0–ICWENR2)	Saf-Relv	*36.5.1.19 on page 739*
0x2EC	Reserved		
0x2F0–0x2F8	Internal Channel Analog Watchdog Out of Range Registers 0–2 (ICAWORR0–ICAWORR2)	No-Saf	*36.5.1.20 on page 740*
0x2FC–0x3FC	Reserved		

Table 370. SARADC digital interface register map(Continued)

Offset	Register	Register classification (1)	Location
0x400	Test Channel Interrupt Pending Register (TCIPR)	Saf-Relv	36.5.1.21 on page 741
0x404	Test Channel Interrupt Mask Register (TCIMR)	Saf-Lat	36.5.1.22 on page 742
0x408	Test Channel DMA Select Register (TCDSR)	Saf-Lat	36.5.1.23 on page 742
0x40C	Test Channel Normal Conversion Mask Register (TCNCMR)	Saf-Lat	36.5.1.24 on page 743
0x410	Test Channel Injected Conversion Mask Register (TCJCMR)	Saf-Lat	36.5.1.25 on page 744
0x414–0x420	Test Channel Watchdog Selection Registers 0–3 (TCWSELR0–TCWSELR3)	Saf-Lat	36.5.1.26 on page 745
0x424	Test Channel Watchdog Enable Register (TCWENR)	Saf-Relv	36.5.1.27 on page 746
0x428	Test Channel Analog Watchdog Out of Range Register (TCAWORR)	Saf-Relv	36.5.1.28 on page 747
0x42C	Reserved		
0x430–0x44C	Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7)	Saf-Relv	36.5.1.29 on page 748
0x450–0x4CC	Test Channel Data Registers 96-127 (TCDR96-TCDR127)	Saf-Relv	36.5.1.30 on page 749
0x4D0–0x4FC	Reserved		
0x500	External Channel Decode Signals Delay Register (ECDSDR)	Saf-Relv	36.5.1.31 on page 751
0x504–0x50C	Reserved		
0x510–0x51C	External Channel Interrupt Pending Registers 0–3 (ECIPR0–ECIPR3)	Saf-Relv	36.5.1.32 on page 752
0x520–0x52C	External Channel Interrupt Mask Registers 0–3 (ECIMR0–ECIMR3)	Saf-Relv	36.5.1.33 on page 753
0x530–0x53C	External Channel DMA Select Registers 0–3 (ECDSR0–ECDSR3)	Saf-Relv	36.5.1.34 on page 754
0x540–0x54C	External Channel Normal Conversion Mask Registers 0–3 (ECNCMR0–ECNCMR3)	Saf-Relv	36.5.1.35 on page 755
0x550–0x55C	External Channel Injected Conversion Mask Registers 0–3 (ECJCMR0–ECJCMR3)	Saf-Relv	36.5.1.36 on page 756
0x560–0x59C	External Channel Watchdog Selection Registers 0–15 (ECWSELR0–ECWSELR15)	Saf-Relv	36.5.1.37 on page 757
0x5A0–0x5AC	External Channel Watchdog Enable Registers 0–3 (ECWENR0–ECWENR3)	Saf-Relv	36.5.1.38 on page 758

Table 370. SARADC digital interface register map(Continued)

Offset	Register	Register classification (1)	Location
0x5B0–0x5BC	External Channel Analog Watchdog Out of Range Registers 0–3 (ECAWORR0–ECAWORR3)	Saf-Relv	*36.5.1.39 on page 759*
0x5C0–0x5CC	External Channel Mapping to Internal Channel Registers 0–3 (ECMICR0–ECMICR3)	Saf-Relv	*36.5.1.40 on page 760*
0x5D0–0x7CC	External Channel Data Registers 128-255 (ECDR128-ECDR255)	Saf-Relv	*36.5.1.41 on page 761*

1. No-Saf = Not Safety Relevant, Saf-Relv = Safety Relevant, Saf-Lat = Safety Latent.

36.5.1 Register descriptions

36.5.1.1 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(1)	NEDGESEL(1)		JSTART	JTRGEN(1)	JEDGESEL(1)		JTRGSEQ(1)	0	CTUEN(2)	CTU_MODE(2)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	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	EDCSELF	PWDN
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. These bits may be written, but will have no effect on the device.
2. This bit is not available on this device and is thus reserved.

Figure 329. Main Configuration Register (MCR)

Table 371. 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)).

Table 371. MCR field descriptions(Continued)

Field	Description
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.
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.
14–15	Reserved
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 371. 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
30 EDCSELF	External Decode Channel Format Select (EDCSELF) This bit selects the format for 3-bit output port ipp_decode_extch used as select line for external decode channel 8:1 muxes. The last three bits of external channel number are passed as is (if value 0) or converted to gray code (if value 1) and passed to this port. 0 Binary format. 1 Gray Code format.
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

36.5.1.2 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 330. Main Status Register (MSR)

Table 372. 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.
15	Reserved
16–23 CHADDR[7:0]	Channel under measure address This status bit is used to signal which channel is under measure.

Table 372. MSR field descriptions(Continued)

Field	Description
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 —

36.5.1.3 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 331. Interrupt Status Register (ISR)

Table 373. ISR field descriptions

Field	Description
0–25	Reserved Write of any value has no effect; read value is always 0.
26–27	Reserved
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.

Table 373. ISR field descriptions(Continued)

Field	Description
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.

36.5.1.4 Internal Channel Interrupt Pending Registers 0–2 (ICIPR0–ICIPR2)

The interrupt channel register to channel association is described in *Table 375*.

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 332. Internal Channel Interrupt Pending Registers (ICIPR0–ICIPR2)

Table 374. 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 375. ICIPR0–2 Registers to channel association

Register	Register bits 31:0
ICIPR0	EOC_CH[31:0]

Table 375. ICIPR0–2 Registers to channel association(Continued)

Register	Register bits 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.

36.5.1.5 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 functionally used, but this bit can be written by software.
2. Only available with cross-triggering unit (CTU) feature.

Figure 333. Interrupt Mask Register (IMR)

Table 376. IMR field descriptions

Field	Description
0–26	Reserved Write of any value has no effect; read value is always 0.
27	Reserved

Table 376. IMR field descriptions(Continued)

Field	Description
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.

36.5.1.6 Internal Channel Interrupt Mask Registers 0–2 (ICIMR0–ICIMR2)

The interrupt mask register to channel association is described in *Table 378*.

Offset 0x024–0x02C 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 334. Internal Channel Interrupt Mask Registers (ICIMR0–ICIMR2)

Table 377. 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 378. 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.

36.5.1.7 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 335. Watchdog Threshold Interrupt Status Register (WTISR)

Table 379. 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.

36.5.1.8 Watchdog Threshold Interrupt Mask Register (WTIMR)

This register gives the interrupt mask information for the 16 possible upper and 16 possible lower threshold limits which can be selected for each channel.

Offset 0x034 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSKWDG15H	MSKWDG15L	MSKWDG14H	MSKWDG14L	MSKWDG13H	MSKWDG13L	MSKWDG12H	MSKWDG12L	MSKWDG11H	MSKWDG11L	MSKWDG10H	MSKWDG10L	MSKWDG9H	MSKWDG9L	MSKWDG8H	MSKWDG8L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 336. Watchdog Threshold Interrupt Mask Register (WTIMR)

Table 380. 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.

36.5.1.9 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 337. DMA Enable Register (DMAE)

Table 381. 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

36.5.1.10 Internal Channel DMA Select Registers 0–2 (ICDSR0–ICDSR2)

The DMA select registers to channel association is described in *Table 383*.

Offset 0x044–0x04C 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 338. Internal Channel DMA Select Register (ICDSR0–ICDSR2)

Table 382. 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 383. 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]

36.5.1.11 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 339. Watchdog Threshold Registers 0–3 (WTHRHLR0–WTHRHLR3)

Table 384. 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.

36.5.1.12 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 340. Conversion Timing Registers 0–3 (CTR0–CTRL3)

Table 385. 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). 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). Refer to device datasheet for the minimum duration required for sampling phase.

36.5.1.13 Internal Channel Normal Conversion Mask Registers 0–2 (ICNCMR0–ICNCMR2)

The normal conversion mask registers to channel association is described in *Table 387*.

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 341. Internal Channel Normal Conversion Mask Registers (ICNCMR0–ICNCMR2)

Table 386. 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 387. 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.

36.5.1.14 Internal Channel Injected Conversion Mask Registers 0–2 (ICJCMR0–ICJCMR2)

The injected conversion mask registers to channel association is described in *Table 389*.

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 342. Internal Channel Injected Conversion Mask Registers (ICJCMR0–ICJCMR2)

Table 388. 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 389. 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.

36.5.1.15 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 343. Power Down Exit Delay Register (PDEDR)

Table 390. 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)].

36.5.1.16 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	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 344. Internal Channel Data Registers (ICDR0–ICDR95)

Table 391. ICDR0–ICDR95 field descriptions

Field	Description
0	Reserved Write of any value has no effect; read value is always 0.
1 REFSEL	Reference selection This bit can be used to select the reference voltage for channel conversion. 0 Selects the default reference (e.g., : 5 V) 1 Selects the alternate reference (e.g., : 2 V)
2–3	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
8–11	Reserved Write of any value has no effect; read value is always 0.
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 391. 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.

The REFSEL bit implementation is dependent on parameter vector altref_exists_ch[95:0].

36.5.1.17 Watchdog Threshold Registers 4–15 (WTHRHLR4–WTHRHLR15)

Offset 0x280–0x2AC 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 345. Watchdog Threshold Registers 4–15 (WTHRHLR4–WTHRHLR15)

Table 392. WTHRHLR4–WTHRHLR15 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. In case num_watchdog is zero, these registers are not implemented.

The length of the threshold field depends on the generic resolution.

36.5.1.18 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 394*.

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 346. Internal Channel Watchdog Select Registers (ICWSELR0–ICWSELR11)

Table 393. 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 394. 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 implemented. For the implemented nibbles, width of WSEL_CHx field within the nibble is dependent on num_wdg parameter For example 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.

36.5.1.19 Internal Channel Watchdog Enable Registers 0–2 (ICWENR0–ICWENR2)

The Watchdog enable registers to channel association is described in *Table 396*.

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 347. Internal Channel Watchdog Enable Registers (ICWENR0–ICWENR2)

Table 395. 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 396. 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.

36.5.1.20 Internal Channel Analog Watchdog Out of Range Registers 0–2 (ICAWORR0–ICAWORR2)

The analog watchdog registers to channel association is described in *Table 398*.

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 348. Internal Channel Analog Watchdog Out of Range Registers (ICAWORR0–ICAWORR2)

Table 397. 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 398. 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.

36.5.1.21 Test Channel Interrupt Pending Register (TCIPR)

The interrupt channel register to channel association is described in *Table 400*.

Offset 0x400 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 349. Test Channel Interrupt Pending Register (TCIPR)

Table 399. 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 400. 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.

36.5.1.22 Test Channel Interrupt Mask Register (TCIMR)

The interrupt mask register to channel association is described in *Table 402*.

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 350. Test Channel Interrupt Mask Register (TCIMR)

Table 401. 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 402. 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.

36.5.1.23 Test Channel DMA Select Register (TCDSR)

The DMA select registers to channel association is described in *Table 404*.

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 351. Test Channel DMA Select Register (TCDSR)

Table 403. 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 404. 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.

36.5.1.24 Test Channel Normal Conversion Mask Register (TCNCMR)

The normal conversion mask register to channel association is described in *Table 406*.

Offset 0x40C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 352. Test Channel Normal Conversion Mask Register (TCNCMR)

Table 405. 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 406. 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.

36.5.1.25 Test Channel Injected Conversion Mask Register (TCJCMR)

The injected conversion mask register to channel association is described in *Table 408*.

Offset 0x410 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 353. Test Channel Injected Conversion Mask Register (TCJCMR)

Table 407. 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 408. 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.

36.5.1.26 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 410*.

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 354. Test Channels Watchdog Select Registers 0–3 (TCWSELR0–TCWSELR3)

Table 409. 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 410. 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.

36.5.1.27 Test Channel Watchdog Enable Register (TCWENR)

The Watchdog enable registers to channel association is described in *Table 412*.

Offset 0x424 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 355. Test Channel Watchdog Enable Register (TCWENR)

Table 411. 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 412. 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.

36.5.1.28 Test Channel Analog Watchdog Out of Range Register (TCAWORR)

The analog watchdog registers to channel association is described in *Table 414*.

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 356. Test Channel Analog Watchdog Out of Range Register (TCAWORR)

Table 413. 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 414. 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.

36.5.1.29 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 416*.

Offset 0x430–0x44C Access: User Read/Write

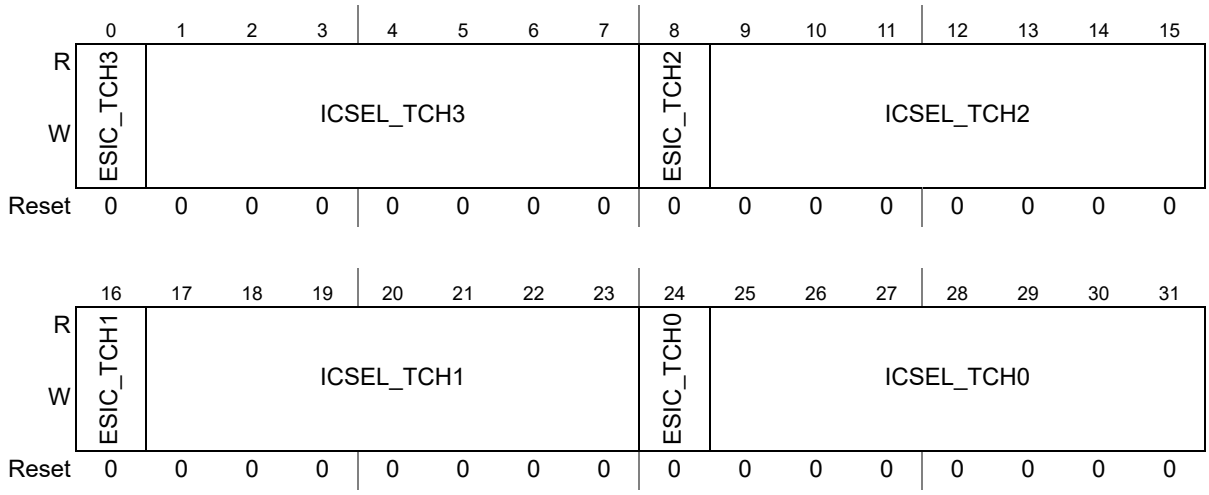


Figure 357. Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7)

For analog test channel assignment, refer to the device configuration or ADC configuration section.

Table 415. 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 416. 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.

36.5.1.30 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	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 358. Test Channel Data Registers (TCDR96–TCDR127)

Table 417. TCDR96–TCDR127 field descriptions

Field	Description
0	Reserved Write of any value has no effect; read value is always 0.
1 REFSEL	Reference selection This bit can be used to select the reference voltage for channel conversion. 0 Selects the default reference (for e.g., : 5V) 1 Selects the alternate reference (for e.g., : 2V)
2–3	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
8–11	Reserved Write of any value has no effect; read value is always 0.
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.

Table 417. TCDR96–TCDR127 field descriptions(Continued)

Field	Description
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].

36.5.1.31 External Channel Decode Signals Delay Register (ECDSDR)

Offset 0x500 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	DSD											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 359. External Channel Decode Signals Delay Register (ECDSDR)

Table 418. ECDSDR field descriptions

Field	Description
0–19	Reserved Write of any value has no effect; read value is always 0.
20–31 DSD	Decode signals delay This bitfield defines the delay between the external decode signals and the start of the sampling phase. It is used to take into account of the settling time required for the external mux. The decode signal delay is calculated as (DSD X 1/Frequency of SARADC clock).

If parameter num_extch is all 0's, the ECDSDR is not implemented and treated as reserved space. Any attempt to access this space generates a transfer error.

36.5.1.32 External Channel Interrupt Pending Registers 0–3 (ECIPR0–ECIPR3)

The interrupt channel register to channel association is described in *Table 420*

Offset 0x510–0x51C 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 360. External Channel Interrupt Pending Registers (ECIPR0–ECIPR3)

Table 419. ECIPR0–ECIPR3 register 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 420. ECIPR0–ECIPR3 Registers to Channel Association

Register	Register bits 31:0
ECIPR0	EOC_CH[159:128]
ECIPR1	EOC_CH[191:160]
ECIPR2	EOC_CH[223:192]
ECIPR3	EOC_CH[255:224]

Parameter num_extch vector has positional inference for registers ECIPR0–ECIPR3 such that:

- num_extch(i) (i = 0–31) corresponds to ECIPR0 bit[0:31].
- num_extch(i) (i = 32–63) corresponds to ECIPR1 bit[0:31].
- num_extch(i) (i = 64–95) corresponds to ECIPR2 bit[0:31].
- num_extch(i) (i = 96–127) corresponds to ECIPR3 bit[0:31].

If any of the num_extch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If any or all num_extch[31:0]/[63:32]/[95:64]/[127:96] is all 0’s, access to corresponding register(s) ECIPR0/1/2/3 generates a transfer error.

36.5.1.33 External Channel Interrupt Mask Registers 0–3 (ECIMR0–ECIMR3)

The interrupt mask register to channel association is described in *Table 422*.

Offset 0x520–0x52C 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 361. External Channel Interrupt Mask Registers (ECIMR0–ECIMR3)

Table 421. ECIMR0–ECIMR3 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 422. External Channel Interrupt Mask Registers to Channel Association

Register	Register bits 31:0
ECIMR0	IM_CH[159:128]
ECIMR1	IM_CH[191:160]
ECIMR2	IM_CH[223:192]
ECIMR3	IM_CH[255:224]

Parameter num_extch vector has positional inference for registers ECIMR0–ECIMR3 such that

- num_extch(i) (i = 0–31) corresponds to ECIMR0 bit[0:31]
- num_extch(i) (i = 32–63) corresponds to ECIMR1 bit[0:31]
- num_extch(i) (i = 64–95) corresponds to ECIMR2 bit[0:31]
- num_extch(i) (i = 96–127) corresponds to ECIMR3 bit[0:31].

If any of the num_extch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If any or all of num_extch[31:0]/[63:32]/[95:64]/[127:96] is all 0’s, access to corresponding register(s) ECIMR0/1/2/3 generates a transfer error.

36.5.1.34 External Channel DMA Select Registers 0–3 (ECDSR0–ECDSR3)

The DMA select registers to channel association is described in *Table 424*.

Offset 0x530–0x53C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 362. External Channel DMA Select Registers (ECDSR0–ECDSR3)

Table 423. ECDSR0–ECDSR3 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.

Note: In case either dma_exist generic parameter is zero or num_extch is zero, these registers are not implemented.

Table 424. DMA Select Registers to Channel Association

Register	Register bits 31:0
ECDSR0	DS_CH[159:128]
ECDSR1	DS_CH[191:160]
ECDSR2	DS_CH[223:192]
ECDSR3	DS_CH[255:224]

Parameter num_extch vector has positional inference for registers ECDSR0–ECDSR3, such that

- num_extch(i) (i = 0–31) corresponds to ECDSR0 bit[0:31].
- num_extch(i) (i = 32–63) corresponds to ECDSR1 bit[0:31].
- num_extch(i) (i = 64–95) corresponds to ECDSR2 bit[0:31].
- num_extch(i) (i = 96–127) corresponds to ECDSR3 bit[0:31].

If any of the num_extch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If any or all of num_extch[31:0]/[63:32]/[95:64]/[127:96] is all 0’s, the access to corresponding register(s) ECDSR0/1/2/3 generates a transfer error.

36.5.1.35 External Channel Normal Conversion Mask Registers 0–3 (ECNCMR0–ECNCMR3)

The normal conversion mask registers to channel association is described in *Table 426*.

Offset 0x540–0x54C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 363. External Channel Normal Conversion Mask Registers (IECNCMR0–ECNCMR3)

Table 425. ECNCMR0–ECNCMR3 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 426. External Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 31:0
ECNCMR0	NCE_CH[159:128]
ECNCMR1	NCE_CH[191:160]
ECNCMR2	NCE_CH[223:192]
ECNCMR3	NCE_CH[255:224]

Parameter num_extch vector has positional inference for registers ECNCMR0–3 such that:

- num_extch(i) (i = 0–31) corresponds to ECNCMR0 bit[0:31]
- num_extch(i) (i = 32–63) corresponds to ECNCMR1 bit[0:31]
- num_extch(i) (i = 64–95) corresponds to ECNCMR2 bit[0:31]
- num_extch(i) (i = 96–127) corresponds to ECNCMR3 bit[0:31].

If any of the num_extch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If any or all of num_extch[31:0]/[63:32]/[95:64]/[127:96] is all 0’s, the access to corresponding register(s) ECNCMR0/1/2/3 generates a transfer error.

36.5.1.36 External Channel Injected Conversion Mask Registers 0–3 (ECJCMR0–3)

The injected conversion mask registers to channel association is described in *Table 428*.

Offset 0x550–0x55C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 364. External Channel Injected Conversion Mask Registers (ECJCMR0–3)

Table 427. ECJCMR0–3 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 428. External Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 31:0
ECJCMR0	JCE_CH[159:128]
ECJCMR1	JCE_CH[191:160]
ECJCMR2	JCE_CH[223:192]
ECJCMR3	JCE_CH[255:224]

Parameter num_extch vector has positional inference for registers ECJCMR0–3 such that

- num_extch(i) (i = 0–31) corresponds to ECJCMR0 bit[0:31]
- num_extch(i) (i = 32–63) corresponds to ECJCMR1 bit[0:31]
- num_extch(i) (i = 64–95) corresponds to ECJCMR2 bit[0:31]
- num_extch(i) (i = 96–127) corresponds to ECJCMR3 bit[0:31].

If any of the num_extch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If any or all of num_extch[31:0]/[63:32]/[95:64]/[127:96] are all 0’s, the access to the corresponding register(s) ECJCMR0/1/2/3 generates a transfer error.

36.5.1.37 External Channel Watchdog Select Registers (ECWSELR0–15)

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 430*.

Offset 0x0560–0x059C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WSEL_CH135				WSEL_CH134				WSEL_CH133				WSEL_CH132			
Reset	0	0	0	0	0	0	0	0	0	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_CH131				WSEL_CH130				WSEL_CH129				WSEL_CH128			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 365. External Channel Watchdog Select Registers (ECWSELR0–15)

Table 429. ECWSELR0–15 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 430. External Channel Watchdog Enable Registers to Channel Association

Register	Register bits 31:0
ECWSELRx (x = 0 to 15)	WSEL_CH[(8*x)+135] .. WSEL_CH[(8*x)+128]]

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, then only 1 bit is present.

Each such nibble (WSEL_CHx) has positional correspondence to num_extch parameter (each channel has a 1:1 nibble correspondence), so each word ECWSELRx corresponds to eight external channels, thus requiring a total of sixteen 32-bit registers to cover all 128 external channels.

If value of any num_extch(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.

36.5.1.38 External Channel Watchdog Enable Registers 0–3 (ECWENR0–ECWENR3)

The Watchdog enable registers to channel association is described in *Table 432*.

Offset 0x5A0–0x5AC Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 366. External Channel Watchdog Enable Registers (ECWENR0–ECWENR3)

Table 431. ECWENR0–ECWENR3 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 432. External Channel Watchdog Enable Registers to Channel Association

Register	Register bits 31:0
ECWER0	WEN_CH[159:128]
ECWER1	WEN_CH[191:160]
ECWER2	WEN_CH[223:192]
ECWER3	WEN_CH[255:224]

Parameter num_extch vector has positional inference for registers ECWER0–ECWER3, such that

- num_extch(i) (i = 0–31) corresponds to ECWER0 bit[0:31].
- num_extch(i) (i = 32–63) corresponds to ECWER1 bit[0:31.]
- num_extch(i) (i = 64–95) corresponds to ECWER2 bit[0:31].
- num_extch(i) (i = 96–127) corresponds to ECWER3 bit[0:31].

If any num_extch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If any or all of num_extch[31:0]/[63:32]/[95:64]/[127:96] are ‘0’, the access to corresponding register(s) ECWER0/1/2/3 generates a transfer error.

36.5.1.39 External Channel Analog Watchdog Out of Range Registers 0–3 (ECAWORR0–ECAWORR3)

The analog watchdog registers to channel association is described in *Table 434*.

Offset 0x05B0–0x05BC 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 367. External Channel Analog Watchdog Out of Range Registers (ECAWORR0–ECAWORR3)

Table 433. ECAWORR0–ECAWORR3 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 434. External Channel Analog Watchdog Registers to Channel Association

Register	Register bits 31:0
ECAWORR0	AWOR_CH[159:128]
ECAWORR1	AWOR_CH[191:160]
ECAWORR2	AWOR_CH[223:192]
ECAWORR3	AWOR_CH[255:224]

Parameter num_extch vector has positional inference for registers ECAWORR0–ECAWORR3 such that:

- num_extch(i) (i = 0–31) corresponds to ECAWORR0 bit[0:31].
- num_extch(i) (i = 32–63) corresponds to ECAWORR1 bit[0:31].
- num_extch(i) (i = 64–95) corresponds to ECAWOERR2 bit[0:31].
- num_extch(i) (i = 96–127) corresponds to ECAWORR3 bit[0:31].

If any num_extch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If any or all of num_extch[31:0]/[63:32]/[95:64]/[127:96] are ‘0’, access to corresponding register(s) ECAWORR0/1/2/3 generates a transfer error.

36.5.1.40 External Channel Mapping to Internal Channel Registers 0–3 (ECMICR0–ECMICR3)

Each set of eight external channels can be mapped with any internal analog channel using the configuration programmed in these registers.

Offset 0x5C0–0x05CC Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	ICSEL_ECH152_159							0	ICSEL_ECH144_151						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	ICSEL_ECH136_143							0	ICSEL_ECH128_135						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 368. External Channel Mapping to Internal Channel Registers 0–3 (ECMICR0–ECMICR3)

Table 435. ECMICR0–ECMICR3 field descriptions

Field	Description
Bit 8x+7	Reserved Write of any value has no effect; read value is always 0.
Bit 8x+6 : 8x ICSEL_ECH	Internal channel selection for external channels (128 + 8x) to (135+8*x) where x = 0,1,2..16

Table 436. Test Channel Short Registers to Channel Association

Register	Register bits 31:0
ECMICRx (x = 0 to 3)	ICSEL_ECH[128 + (32*x+31)]_[128 + (32*x+24)] ICSEL_ECH[128 + (32*x + 23)]_ [128 + (32*x+16)], ICSEL_ECH[128 + (32*x+15)]_[128 + (32*x+8)] ICSEL_ECH[128 + (32*x + 7)]_ [128 + 32*x],

Parameter num_extch vector has positional inference for registers ECMICR0–ECMICR3 such that:

- num_extch(i) (i = 0–31) corresponds to ECMICR0.
- num_extch(i) (i = 32–63) corresponds to ECMICR1.
- num_extch(i) (i = 64–95) corresponds to ECMICR2.
- num_extch(i) (i = 96–127) corresponds to ECMICR3.

If any or all of num_extch[31:0]/[63:32]/[95:64]/[127:96] parameter slices are all ‘0’s, the access to corresponding register(s) ECMICR0/1/2/3 generates a transfer error.

36.5.1.41 External Channel Data Registers 128–255 (ECDR128–ECDR255)

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 0x5D0–0x7CC — Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 369. External Channel Data Registers (ECDR128–ECDR255)

Table 437. ECDR128–ECDR255 field descriptions

Field	Description
0	Reserved. Write of any value has no effect; read value is always 0.
1 REFSEL	Reference selection. This bit can be used to select the reference voltage for channel conversion. 0 Selects the default reference (for e.g., : 5 V) 1 Selects the alternate reference (for e.g., : 2 V)
2–3	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
8–11	Reserved Write of any value has no effect; read value is always 0.
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 437. ECDR128–ECDR255 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*. Each ECDRx corresponds to an external channel inferred through generic num_extch and if the value of any num_extch(i) is '0', the corresponding ECDRx is not implemented. Any access on this space generates transfer error.

This register should be accessed 32-bit R/W.

The REFSEL bit implementation is dependent on parameter vector altref_exists_ch[255:128].

36.6 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 hardmacro cell. Here each possible trigger source is described with the number of clock cycles required for different operations until trigger of adc_start. This table does not take into account any software delay involved in programming or polling of any bit in order to measure conversion timing.

Table 438. 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 438. 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.

36.7 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.
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, overwrite enable e.t.c. in MCR register.
4. Program the conversion mask registers({I,T,E}C{N,J}CMR) for channels to be converted. If required, enable and select watchdog monitors({I,T,E}CWENR, {I,T,E}CWSELR) as applicable for each channel, program the watchdog threshold values (WTHRHLR) for each watchdog monitor.
5. If required, Enable and select the External to Internal channel mapping by programming the ECMICR registers, Test to internal channel mapping by programming the TCCAPR registers.
6. Program the required Interrupt/DMA mask registers (DMAE, {I,T,E}CDSR, IMR, {I,T,E}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,E}CDR registers.
8. 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,E}CDR registers.

37 Decimation Filter

37.1 Introduction

This chapter describes the Decimation Filter block.

37.1.1 Overview

The decimation filter is a dedicated hardware block, designed to decimate fixed point sample conversion results. An interface is provided for use by the CPU, allowing setup of the filter parameters and read/write of the configuration registers and filter samples.

The Decimation Filter receives data samples from the master blocks (CPU) in the RX sub-block. Each sample arrives at the decimation filter with an identifier tag and associated commands. The input information is decoded by the RX and control logic sub-blocks. When receiving a filtering command, the data is transferred to the filter tap register's sub-block and is processed by the filter using the MAC (Multiply and Accumulate) unit, the coefficient register, and the control logic sub-blocks. The result is then returned to the master block by the TX sub-block, accompanied by the corresponding tag information that provides an address for the data.

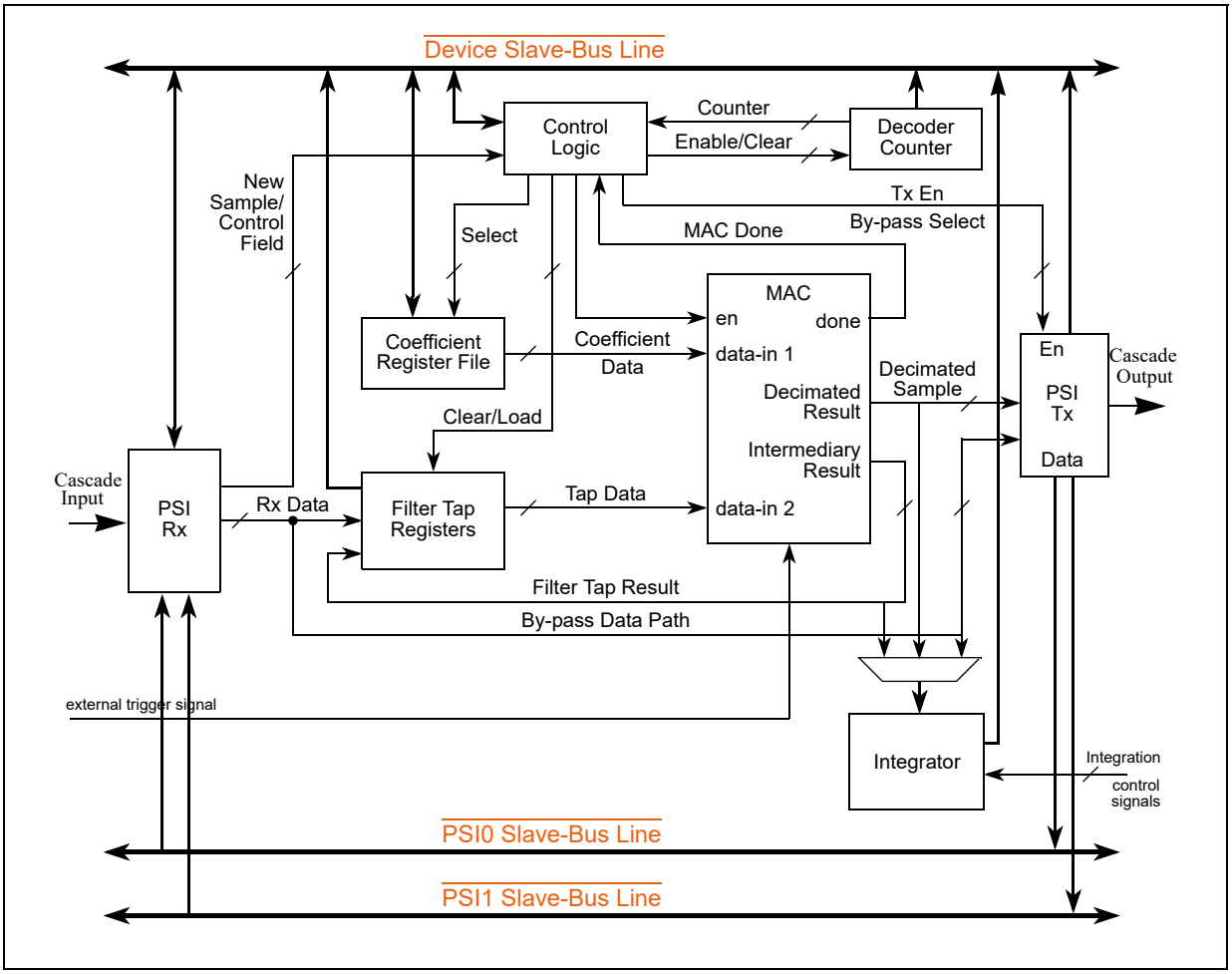
The decimation filter functions in a standalone mode. In this mode, the input data is supplied and the output results are read by the chip core processor (CPU) using status and interrupt signals or DMA requests.

An integrator unit independently accumulates the values of filter outputs. The integration can be restricted to time windows defined by hardware signals or software.

All signals in the interface are generated in the system clock domain.

Figure 370 shows the block diagram for the decimation filter.

Figure 370. Decimation Filter Block Diagram



37.1.2 Features

The Decimation Filter block includes these distinctive features:

- Selectable 4th order IIR filter, or an 8th order FIR filter
  - Input/output with 16-bit (fixed point) two's complement signed values
  - Internal taps with 16-bit (feed-forward portion of first IIR) and 24-bit (feedback portion) resolutions (fixed point) for two's complement signed value
  - 24-bit programmable filter coefficients (fixed point) for two's complement signed value
  - MAC unit with 51-bit fixed point accumulator
  - Convergent rounding methodology
  - Two's complement overflow or saturation selection
  - 58 clock cycles to process the input
- Input and output buffers with DMA capability
- Slave-bus interface to device
- Filter taps access for debug
- Filter initialization (flush) and stabilization (prefill) commands
- Decimation controlled by an internal counter
- Integrator unit accumulates filter output values, signaled or absolute, with 32-bit resolution. The integrator can be controlled by software or hardware signals.

37.1.3 Modes of operation

This section describes the operation modes of the Decimation Filter. The modes are selected using the following DECFILTER_MCR register fields: MDIS, FREN, FRZ, ISEL (see *Section 37.3.2.1: Decimation Filter Module Configuration Register (DECFILTER_MCR)*).

The mode selection is summarized in *Table 439*

Table 439. Operation Mode Selection

Mode	MDIS	FREN, FRZ	ISEL
Standalone	0	(0, 0) or (0, 1) or (1, 0)	1
Freeze(1)	0	1, 1	X
Low Power	1	X	X
Reserved	All other combinations		

1. Freeze mode can also be activated from outside the Decimation Filter, depending on the MCU, if FREN=1.

37.1.3.1 Standalone mode

In this case, the data is provided by the central processor using the device slave-bus interface or DMA interface signals. Once the data is filtered the decimated result is available

in the Output Buffer register. The filter output is also consumed by a CPU or DMA mastering the same device slave-bus interface. This operation mode can be used to debug the filter stability or to decimate data in System RAM.

37.1.3.2 Low Power mode

Low power mode corresponds to the module disable mode or stop mode. In the module disable mode the filter input is disabled and it is not possible to enter Freeze mode. The system clock is stopped. And in stop mode, the system clock is also stopped.

37.1.3.3 Freeze mode

This mode is also known as debug mode. All filter action is frozen, either through software or by the hardware SoC debug request signal. If a freeze request comes when the filter is processing an input, it enters freeze mode only after the processing finishes.

37.2 External signal description

Note: The Decimation Filter does not provide metastability protection or filtering for these signals.

37.2.1 Integrator enable signal

This signal is used to enable the hardware integrator. For more details, see *Section 37.4.13.4: Integrator enabling and halting*.

37.2.2 Integrator halt signal

This signal is used to halt the hardware integrator. For more details, see *Section 37.4.13.4: Integrator enabling and halting*.

37.2.3 Integrator reset signal

This signal is used to reset the hardware integrator. For more details, see *Section 37.4.13.3: Integrator reset*.

37.2.4 Integrator output request signal

This signal is used to update the integrator output result. For more details, see *Section 37.4.13.2: Integrator outputs*.

37.3 Memory map and register definition

This section provides the memory maps and detailed descriptions of all registers. Accesses to reserved areas of the memory map can return a bus error, depending on the device integration.

This module communicates with the slave-bus line that is related to the device integration. Below the memory map is described.

37.3.1 Decimation filter device memory map

The addresses of the Decimation Filter registers are specified as offsets from the module's base address, described in *Table 440*. The registers allocated in this memory map are sufficient for a 4th order IIR filter implementation.

Table 440. Block memory map

Address offset	Register	Access	Reset Value	Section/Page
0x000	DECFILTER_MCR — Module Configuration Register	R/W	—(1)	*37.3.2.1/772*
0x004	DECFILTER_MSR — Module Status Register	R/W	0x0000_0000	*37.3.2.2/776*
0x008	DECFILTER_MXCR — Module Extended Config.Register	R/W	0x0000_0000	*37.3.2.3/779*
0x00C	DECFILTER_MXSR — Module Extended Status Register	R/W	0x0000_0000	*37.3.2.4/781*
0x010	DECFILTER_IB — Interface Input Buffer	R/W	0x0000_0000	*37.3.2.5/784*
0x014	DECFILTER_OB — Interface Output Buffer	R	0x0000_0000	*37.3.2.6/785*
0x018–0x01F	Reserved			
0x020	DECFILTER_COEF0 — Filter Coefficient 0	R/W	0x0000_0000	*37.3.2.7/785*
0x024	DECFILTER_COEF1 — Filter Coefficient 1	R/W	0x0000_0000	
0x028	DECFILTER_COEF2 — Filter Coefficient 2	R/W	0x0000_0000	
0x02C	DECFILTER_COEF3 — Filter Coefficient 3	R/W	0x0000_0000	
0x030	DECFILTER_COEF4 — Filter Coefficient 4	R/W	0x0000_0000	
0x034	DECFILTER_COEF5 — Filter Coefficient 5	R/W	0x0000_0000	
0x038	DECFILTER_COEF6 — Filter Coefficient 6	R/W	0x0000_0000	
0x03C	DECFILTER_COEF7 — Filter Coefficient 7	R/W	0x0000_0000	
0x040	DECFILTER_COEF8 — Filter Coefficient 8	R/W	0x0000_0000	
0x044–0x077	Reserved			

Table 440. Block memory map(Continued)

Address offset	Register	Access	Reset Value	Section/Page
0x078	DECFILTER_TAP0 — Filter TAP(2) 0 Register	R	0x0000_0000	37.3.2.8/786
0x07C	DECFILTER_TAP1 — Filter TAP 1 Register	R	0x0000_0000	
0x080	DECFILTER_TAP2 — Filter TAP 2 Register	R	0x0000_0000	
0x084	DECFILTER_TAP3 — Filter TAP 3 Register	R	0x0000_0000	
0x088	DECFILTER_TAP4 — Filter TAP 4 Register	R	0x0000_0000	
0x08C	DECFILTER_TAP5 — Filter TAP 5 Register	R	0x0000_0000	
0x090	DECFILTER_TAP6 — Filter TAP 6 Register	R	0x0000_0000	
0x094	DECFILTER_TAP7 — Filter TAP 7 Register	R	0x0000_0000	
0x098–0x0CF	Reserved			
0x0D0	Reserved			
0x0D4–0x0DF	Reserved			
0x0E0	DECFILTER_FINTVAL — Final Integr. Value Register	R	0x0000_0000	37.3.2.9/787
0x0E4	DECFILTER_FINTCNT — Final Integr. Count Register	R	0x0000_0000	37.3.2.10/788
0x0E8	DECFILTER_CINTVAL — Current Integr. Value Register	R	0x0000_0000	37.3.2.11/788
0x0EC	DECFILTER_CINTCNT — Current Integr. Count Register	R	0x0000_0000	37.3.2.12/789
0x0F0–0x1FF	Reserved			

1. See register description.
2. The TAP register stores, on each filter node, the input sample data and, for the IIR type, the filter intermediary results.

37.3.2 Decimation filter register descriptions

All registers are 32-bit wide.

37.3.2.1 Decimation Filter Module Configuration Register (DECFILTER_MCR)

The Decimation Filter module configuration register provides configuration control bits for the Decimation Filter internal logic.

Note: One must not modify this register contents when the status bit BSY is set, except for fields FREN, FRZ and IDIS. To guarantee that BSY does not set during the read-modify-write operation, set IDIS=1 and wait for BSY=0 beforehand.

Address: DECFILTER_BASE + 0x000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MDIS	FREN	0	FRZ	0	0	0	IDEN	ODEN	ERREN	0	FTYPE[1:0]		0	SCAL[1:0]	
W					SRES											
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	IDIS	SAT	ISEL	0	DEC_RATE[3:0]				SDIE	DSEL	IBIE	OBIE	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset value is defined by the MDIS_DEFAULT parameter value.

Figure 371. Decimation Filter Module Configuration Register (DECFILTER_MCR)

Table 441. DECFILTER_MCR field descriptions

Field	Description
0 MDIS	Module Disable The MDIS bit puts the Decimation Filter in low power mode. Communication through the slave-bus Interface is ignored in this mode. Writes to the configuration register are allowed with the exception of writes to the FREN and SRES bits, which are ignored. Writes to the Coefficient registers are also allowed. The Decimation Filter cannot enter Freeze mode once in disable mode. Once the module is disabled it no longer receives the system clock. 1 Low Power Mode 0 Normal Mode
1 FREN	Freeze Enable The FREN bit enables the Decimation Filter to enter freeze mode if the SoC debug request signal or the FRZ bit is asserted. See *Section 37.4.12: Freeze mode description*, for more details. 1 Decimation Filter Freeze mode enabled 0 Decimation Filter Freeze mode disabled
2	Reserved
3 FRZ	Freeze Mode The FRZ bit controls the freeze mode of the Decimation Filter. For this bit to take effect the FREN freeze enable bit also needs to be asserted. While in freeze mode the MAC operations are halted. See *Section 37.4.12: Freeze mode description*, for more details. 1 Decimation Filter in Freeze Mode 0 Decimation Filter in Normal Mode
4 SRES	Software-reset bit The SRES is a self-negated bit which provides the CPU with the capability to initialize the Decimation Filter through the slave-bus interface. This bit always reads as zero. See *Section 37.4.9: Soft reset command description*, for more details. 1 Software-Reset 0 No action
5-6	Reserved

Table 441. DECFILTER_MCR field descriptions(Continued)

Field	Description
7 IDEN	Input Data Interrupt Enable The IDEN bit enables the Decimation Filter to generate interrupt requests on all new input data written to the Interface Input Buffer register or Input/Output Buffers register. 1 Input Data Interrupt Enabled 0 Input Data Interrupt Disabled
8 ODEN	Output Data Interrupt Enable The ODEN bit enables the Decimation Filter to generate interrupt requests on all new data written to the filter Output buffer. 1 Output Data Interrupt Enabled 0 Output Data Interrupt Disabled
9 ERREN	Error Interrupt Enable The ERREN bit enables the Decimation Filter to generate interrupt requests based on the assertion of the DECFILTER_MSR register error flags OVF, DIVR, SVR, OVR or IVR. 1 Error Interrupts Enabled 0 Error Interrupts Disabled
10	Reserved
11-12 FTYPE[1:0]	Filter Type Selection bits The FTYPE[1:0] bits select the filter type according to the following values: 00 Filter Bypass 01 IIR Filter - 1 x 4th order 10 FIR Filter - 1 x 8th order 11 reserved **Note:** In Bypass configuration the filter is disabled. **Note:** Bypass must not be configured when in cascade mode (see field CASCD).
13	Reserved
14-15 SCAL[1:0]	Filter Scaling Factor The SCAL[1:0] bit field selects the scaling factor used by the filter algorithm according to the following values: 00 Scaling Factor = 1 01 Scaling Factor = 4 10 Scaling Factor = 8 11 Scaling Factor = 16
16 IDIS	Input Disable The IDIS bit disables the block input, so that writes to the input buffer have no effect (from the device slave-bus), and input DMA or interrupt requests are not issued. 0 Input disabled 1 Input enabled **Note:** IDIS resets in 1 (hardware reset only), and the module configuration procedures for previous versions (without IDIS bit) are upward compatible provided that the DECFILTER_MCR register is written with 0 into the IDIS position (previously reserved).

Table 441. DECFILTER_MCR field descriptions(Continued)

Field	Description
17 SAT	Saturation Enable The SAT bit enables the saturation of the filter output. See *Section 37.4.6.2: Saturation*, for more details. 0 Enable Saturation 1 Disable Saturation
18 ISEL	Input Selection The ISEL bit selects the source of input data to the Filter. The data source is the CPU/DMA on the device slave-bus interface. Each device slave-bus write to the Interface Input Buffer register or DMA transfer to the input buffer is interpreted as a new sample to be processed by the filter. 0 Reserved 1 Filter input from the device slave-bus interface **Note:**
19	Reserved
20-23 DEC_RATE[3:0]	Decimation Rate Selection The DEC_RATE[3:0] field selects the decimation rate used by the Decimation Filter. The decimation rate defines the number of data samples from the master block that is required to generate one decimated result in the Decimation Filter output. 0000 No Decimation: one filter output for each sample input 0000–1111 One filter output for each (DEC_RATE+1) sample inputs
24 SDIE	Integrator Data Interrupt Enable The SDIE field enables output buffer interrupts due to integrator data result being ready (at registers DECFILTER_FINTVAL and DECFILTER_FINTCNT): 0 Integration ready does not cause an output interrupt. 1 Integration ready causes an output interrupt
25 DSEL	DMA Selection The DSEL bit determines whether the data transfers — to the input buffer (write to) and from the output buffer (read from) — are performed by DMA requests or by interrupt requests . 0 Interrupt requests are generated 1 DMA requests are generated
26 IBIE	Input Buffer Interrupt Request Enable The IBIE bit enables the Decimation Filter to generate interrupt requests when: – device slave-bus input is selected (ISEL=1) and DSEL=0 when the input buffer is available to receive new data; 0 Input Buffer Interrupt Request Disabled 1 Input Buffer Interrupt Request Enabled
27 OBIE	Output Buffer Interrupt Request Enable The OBIE bit enables the Decimation Filter interrupt requests when outputs are directed to the device slave-bus and DMA is not selected (DSEL=0). 0 Output Buffer Interrupt Request Disabled 1 Output Buffer Interrupt Request Enabled
28-31	Reserved

37.3.2.2 Decimation Filter Module Status Register (DECFILTER_MSR)

Address: DECFILTER_BASE + 0x004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BSY	0	DEC_COUNTER[3:0]				0	0	0	0	0	0	0	0	0	0
W							IDFC	ODFC		IBIC	OBIC			OVFC	OVRC	IVRC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	IDF	ODF	0	IBIF	OBIF	0	0	OVF	OVR	IVR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 372. Decimation Filter Status Register (DECFILTER_MSR)

Table 442. DECFILTER_MCR field descriptions

Field	Description
0 BSY	Decimation Filter Busy indication The BSY bit indicates that the Decimation Filter is processing new input data from the master block in normal mode or from the device core in standalone mode. BSY is not asserted when the filter is disabled (FTYPE = 00). However, the BSY bit is asserted when the soft reset is executed. 0 Decimation Filter Idle 1 Decimation Filter Busy
1	Reserved
2-5 DEC_COUNTER[3:0]	Decimation Counter The DEC_COUNTER[3:0] field indicates the current value of the DEC_COUNTER Decimation Counter (see *Figure 370*), which counts the number of input data samples received by the Decimation Filter. When the value of this counter matches the DEC_RATE[3:0] Configuration Register field, one decimated result is generated and the DEC_COUNTER counter is re-initialized at zero. This register is cleared by a soft reset or a flush command.
6 IDFC	Input Data Flag Clear bit The IDFC bit clears the IDF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears IDF
7 ODFC	Output Data Flag Clear bit The ODFC bit clears the ODF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears ODF
8	Reserved
9 IBIC	Input Buffer Interrupt Request Clear bit The IBIC bit clears the IBIF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears IBIF

Table 442. DECFILTER_MCR field descriptions(Continued)

Field	Description
10 OBIC	Output Buffer Interrupt Request Clear bit The OBIC bit clears the OBIF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears OBIF
11	Reserved
12	Reserved
13 OVFC	OVF Clear bit The OVFC bit clears the OVF Output Overflow bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears OVF
14 OVRC	OVR Clear bit The OVRC bit clears the OVR Output Overrun bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears OVR
15 IVRC	IVR Clear bit The IVRC bit clears the IVR Filter Input Overrun indication bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears IVR
16-21	Reserved
22 IDF	Input Data Flag The IDF bit flag indicates when new data is available at the DECFILTER_IB register or at the DECFILTER_IOB register. This flag generates an Interrupt Request if enabled by the IDEN bit in the Configuration Register. This Flag is cleared by the IDFC Status bit or by a soft reset of the decimation filter. 0 Sample not received 1 New Sample received **Note:** OBS: This flag is not used for read / write requests. It is used only to announce the input data event. For read / write request flag, refer to IBIF.
23 ODF	Output Data Flag The ODF bit flag indicates when a new decimated sample is available at the DECFILTER_OB register or at the DECFILTER_IOB register. This flag generates an Interrupt Request if enabled by the ODEN bit in the Configuration Register. This Flag is cleared by the ODFC Status bit or by a soft reset of the decimation filter. No new Decimated Output Sample available New Decimated Output Sample available **Note:** OBS: This flag is not used for read requests. It is used only to announce the output data event. For read request flag, refer to OBIF.
24	Reserved

Table 442. DECFILTER_MCR field descriptions(Continued)

Field	Description
25 IBIF	Input Buffer Interrupt Request Flag The IBIF bit flag indicates that the input buffer DECFILTER_IB is available to be filled with new data, when Enhanced Debug Monitor is off. In Enhanced Debug Monitor, it indicates the input buffer DECFILTER_IB was filled with a new sample and is ready to be read. IBIF assertion also asserts the interrupt signal when enabled by the IBIE bit in the Configuration Register when DMA is not selected (DSEL=0) and the input buffer requires access from the device slave-bus (ISEL != EDME). This Flag is cleared by the IBIC Status bit or by a soft reset of the decimation filter. 0 No action 1 New Sample is requested (ISEL = 1, EDME=0) or new sample is available in Enhanced Debug Monitor (ISEL=0, EDME=1).
26 OBIF	Output Buffer Interrupt Request Flag The OBIF bit flag indicates that a new decimated sample is available at the DECFILTER_OB register. This flag generates an Interrupt Request if enabled by the OBIE bit in the Configuration Register and with DSEL=0. This Flag is cleared by the OBIC Status bit or by a soft reset of the decimation filter. 0 No new Decimated Output available 1 New Decimated Output available
27	Reserved
28	Reserved
29 OVF	Filter Overflow Flag The OVF bit indicates that an overflow occurred in the filtered sample result. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the OVFC Status bit or by a soft reset of the decimation filter. 0 No overflow 1 Overflow occurred
30 OVR	Output Interface Buffer Overrun The OVR bit indicates that a decimated sample was overwritten by a new sample in the Interface Output Buffer Register. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the OVRC Status bit or by a soft reset of the decimation filter. 0 No Output Overrun 1 Filter Output Overrun occurred
31 IVR	Input Interface Buffer Overrun The IVR bit indicates that a received sample in the Filter Interface Input Register was overwritten by a new sample. This was probably caused by a violation of the Decimation Filter maximum throughput. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the IVRC Status bit or by a soft reset of the decimation filter. 0Input Buffer Overrun did not occur 1Input Buffer Overrun occurred **Note:** IVR does not set due to input register writes when input is disabled (DECFILTER_MCR bit IDIS=1).

37.3.2.3 Decimation Filter Module Extended Configuration Register (DECFILTER_MXCR)

Address : DECFILTER_BASE + 0x008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SDMAE	SSIG	SSAT	SCSAT	0	0	0	0	0	0	0	0	0	0	0	0
W															SRQ	SZRO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SISEL	0	SZROSEL		0	0	SHLTSEL		0	SRQSEL			0	0	SENSEL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 373. Decimation Filter Extended Configuration Register (DECFILTER_MXCR)

Table 443. DECFILTER_MXCR field descriptions

Field	Description
0 SDMAE	Integrator DMA Enable The SDMAE bit enables the DMA request when an integrator output is requested (see *Section 37.4.13.2: Integrator outputs*). 0 integrator DMA request disabled 1 integrator DMA request enabled **Note:** The DMA channel used is the same one used for filter outputs, and any configuration that generates DMA requests from both of those sources is not allowed.
1 SSIG	Integrator Signal operation selection The SSIG bit defines how the filtered data signal is treated for integration: 0 integrator input takes the absolute value of filter output 1 integrator input takes signaled filter output
2 SSAT	Integrator Saturated operation selection The SSAT bit defines how the integrator accumulator behaves in case of an overflow. 0 integrator accumulator holds a modulo 2^{17} value (considering the 15-bit fractional part) on an overflow. 1 integrator accumulator saturates on an overflow **Note:** In saturated operation the overflowed integration sum holds the value 0xFFFFFFFF for absolute integration (SSIG=0), or values 0x7FFFFFFF (positive saturation) and 0x80000000 (negative saturation) for signaled integration (SSIG=1). **Note:** Non-saturated mode is not supported with signaled integration, therefore one must not configure SSIG=1 and SSAT=0.
3 SCSAT	Integrator Counter Saturated operation selection The SCSAT bit defines how the integrator sample counter behaves in case of an overflow. 0 integrator sample counter holds a modulo 2^{32} value on an overflow. 1 integrator sample counter saturates on an overflow, holding a value of 0xFFFFFFFF.
4-13	Reserved

Table 443. DECFILTER_MXCR field descriptions(Continued)

Field	Description
14 SRQ	Integrator Output Request The SRQ bit is used to command the update of the integrator output, reflected in the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT. It may also cause a DMA or interrupt request, depending on the DECFILTER_MCR bit SDIE and DECFILTER_MXCR bit SDMAE. This is a write-only bit, so reads always return 0. For more details, see *Section 37.4.13.2: Integrator outputs*. 0 no integrator output update request 1 requests integrator output update
15 SZRO	Integrator Zero The SZRO bit is used to zero the integrator sum. This is a write-only bit, reads always return 0. For more details, see *Section 37.4.13.3: Integrator reset*. 0 does not zero integrator sum 1 zeroes integrator sum **Note:** If bits SRQ and SZRO are both written 1 at the same time, the integrator is reset only after the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT are updated.
16 SISEL	Integrator Input Selection The SISEL bit selects the input of the integrator. For more details, see *Section 37.4.13.1: Integrator inputs*. 0 decimated filter outputs feed the integrator 1 filter outputs before the decimation feed the integrator
17	Reserved
18-19 SZROSEL	Integrator Zero Control Mode Selection The SZROSEL field defines the use of the integrator zero hardware input signal, according to the parameters below. For more details, see *Section 37.4.13.3: Integrator reset*. 00 hardware integrator zero request disabled 01 integrator zero on toggle of hardware signal 10 integrator zero on rising edge of hardware signal 11 integrator zero on falling edge of hardware signal
20-21	Reserved
22-23 SHLTSEL	Integrator Halt Control Selection The SHLTSEL field defines the integrator halting mechanism, according to the list below. When the integrator is halted, the integration accumulator remains unaltered on filter outputs independently of the enabling selected by SENSEL. For more details, see *Section 37.4.13.4: Integrator enabling and halting*. 00 hardware halt control signal disabled 01 integrator halted, independently of the hardware signal 10 integrator halted when signal is at logical 0 11 integrator halted when signal is at logical 1
24	Reserved

Table 443. DECFILTER_MXCR field descriptions(Continued)

Field	Description
25-27 SRQSEL	Integrator Output Read Request Mode Selection The SRQSEL field defines the use of the integrator output request hardware input signal, according to the parameters below. An integrator output request updates the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT, also causing a DMA or interrupt request. Note that DMA or interrupt requests due to integrator output updates depend on the DECFILTER_MXCR bit SDMAE and DECFILTER_MCR bit SDIE. When continuous output is on, an integrator output request is issued whenever a new filter output is accumulated. For more details, see *Section 37.4.13.2: Integrator outputs*. 000 hardware output request disabled 001 integrator output request on toggle of hardware signal 010 integrator output request on rising edge of hardware signal 011 integrator output request on falling edge of hardware signal 100 reserved 101 continuous output request on, independently of hardware signal 110 continuous output request on when signal is at logical 0 111 continuous output request on when signal is at logical 1
28-29	Reserved
30-31 SENSEL	SENSEL — Integrator Enable Control Selection The SENSEL field defines the integrator enabling mechanism, according to the parameters below. When the integrator is enabled, filter outputs selected by the SISEL bit are added to the integration accumulator. When the integrator is disabled, the integration accumulator remains unaltered on filter outputs. For more details, see *Section 37.4.13.4: Integrator enabling and halting*. 00 integrator disabled, independently of the hardware enable control signal 01 integrator enabled, independently of the hardware signal 10 integrator enabled when signal is at logical 0 11 integrator enabled when signal is at logical 1

37.3.2.4 Decimation Filter Module Extended Status Register (DECFILTER_MXSR)

Address: DECFILTER_BASE + 0x00C 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								SDFC			SSEC	SCEC		SSOVFC	SCOVFC	SVRC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	SDF	0	0	SSE	SCE	0	SSOVF	SCOVF	SVR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 374. Decimation Filter Extended Status Register (DECFILTER_MXSR)

Table 444. DECFILTER_MXSR field descriptions

Field	Description
0-6	Reserved
7 SDFC	Integrator Output Data Flag Clear bit The SDFC bit clears the SDF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SDF
8-9	Reserved
10 SSEC	Integrator Sum Exception Clear bit The SSEC bit clears the SSE flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SSE
11 SCEC	Integrator Count Exception Clear bit The SCEC bit clears the SCE flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SCE
12	Reserved
13 SSOVFC	Integrator Sum Overflow Clear bit The SSOVFC bit clears the SSOVF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SSOVF
14 SCOVFC	Integrator Count Overflow Clear bit The SCOVFC bit clears the SCOVF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SCOVF
15 SVRC	SVR Clear bit The SVRC bit clears the SVR Integrator Data Overrun indication bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SVR
16-22	Reserved
23 SDF	Integrator Data Flag The SDF bit flag indicates when a new integrator result is available at the DECFILTER_FINTVAL register. This flag generates an Interrupt Request if enabled by the SDIE bit in the Configuration Register. This Flag is cleared by the SDFC Status bit or by a soft reset of the decimation filter. 0 no new integrator result available 1 new integrator result available
24-25	Reserved

Table 444. DECFILTER_MXSR field descriptions(Continued)

Field	Description
26 SSE	Integrator Sum Exception flag The SSE bit indicates an exceptional condition of the integrator accumulator. This flag generates an Interrupt Request if enabled by the DECFILTER_MCR bit ERREN, and it is cleared by the SSEC bit or by a soft reset. Integrator exceptions are defined in *Section 37.4.13.5: Integrator exceptions*. 0 No exception in the integrator accumulator. 1 Integrator accumulator exception.
27 SCE	Integrator Count Exception flag The SSE bit indicates an exceptional condition of the integrator counter. This flag generates an Interrupt Request if enabled by the DECFILTER_MCR bit ERREN, and it is cleared by the SCEC bit or by a soft reset. Integrator exceptions are defined in *Section 37.4.13.5: Integrator exceptions*. 0 No exception in the integrator counter. 1 Integrator counter exception.
28	Reserved
29 SSOVF	Integrator Sum Overflow Flag The SSOVF bit flag indicates an overflow of the integrator accumulator. This Flag is cleared by the SSOVFC bit or by a soft reset. 0 No overflow in the integrator sample counter. 1 Integrator sample counter overflowed. **Note:** The SCOVF bit samples the integrator accumulator overflow condition when and only when either registers DECFILTER_FINTVAL or DECFILTER_CINTCNT are updated. Therefore, only one of the register pairs (DECFILTER_FINTVAL/DECFILTER_FINTCNT and DECFILTER_CINTVAL/DECFILTER_CINTCNT) must be used by the application, in order to avoid races.
30 SCOVF	Integrator Count Overflow Flag The SCOVF bit flag indicates an overflow of the internal integrated sample counter. This Flag is cleared by the SCOVFC bit or by a soft reset. 0 No overflow in the integrator sample counter. 1 Integrator sample counter overflowed. **Note:** The SCOVF bit samples the integrator accumulator overflow condition when and only when either registers DECFILTER_FINTVAL or DECFILTER_CINTCNT are updated. Therefore, only one of the register pairs (DECFILTER_FINTVAL/DECFILTER_FINTCNT and DECFILTER_CINTVAL/DECFILTER_CINTCNT) must be used by the application, in order to avoid races.
31 SVR	Integrator Data Overrun The SVR bit indicates that an integration value and count in the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT was overwritten by a new integrator output request and was not read by the CPU or DMA. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the SVRC bit or by a soft reset. 0 Integrator Data Overrun did not occur. 1 Integrator Data Overrun occurred.

37.3.2.5 Decimation Filter Interface Input Buffer register (DECFILTER_IB)

The Input Buffer Register provides access to the Input buffer of the decimation filter when the filter is in the standalone mode of operation. Writes to this register are interpreted as requests to the Decimation Filter to process new sample data.

Address: DECFILTER_BASE + 0x010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	INTAG[3:0]				0	0	0	0	0	0	PREFILL	FLUSH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INPBUF[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 375. Decimation Filter Interface Input Buffer register (DECFILTER_IB)

Table 445. DECFILTER_IB field descriptions

Field	Description
0-2	Reserved
3	Reserved
4-7 INTAG[3:0]	Decimation filter input tag bits The INTAG[3:0] bit field is defined as a selector signal and it is used to identify different destinations for the INBUF[15:0] data.
8-13	Reserved
14 PREFILL	Decimation Filter Prefill/Filter control bit The PREFILL bit selects the Decimation Filter operation mode. For more details, see *Section 37.4.7: Filter prefill control description*. 0 Decimation Filter normal sample 1 Decimation Filter prefill sample
15 FLUSH	Decimation Filter Flush control bit Assertion of the FLUSH bit initializes the Decimation Filter to a initial state, as defined in *Section 37.4.8: Flush command description*. This bit is self negated and it is cleared only when the data is read and the flush is executed. 0 no flush request 1 flush request
16-31 INPBUFF[15:0]	Input Buffer Data The INPBUF[15:0] bit field carries the sample data to be filtered. This data buffer can be written from the device slave-bus interface. See *Section 37.4.3: Input buffer description*, for more details.

37.3.2.6 Decimation Filter Interface Output Buffer register (DECFILTER_OB)

Address: DECFILTER_BASE + 0x014 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	OUTTAG[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUTBUF[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 376. Decimation Filter Interface Output Buffer register (DECFILTER_OB)

Table 446. DECFILTER_OB field descriptions

Field	Description
0-10	Reserved
11	Reserved
12-15 OUTTAG[3:0]	Decimation filter output tag bits The OUTTAG[3:0] bit field is defined as a selector signal and it is used to identify different destinations for the OUTBUF[15:0] data.
16-31 OUTBUFF[15:0]	Output Buffer Data The OUTPBUF[15:0] bit field is the result data in the decimation filter Output Buffer. It represents a fixed point signed number in two's complement format and is updated only when a decimated result is ready to be transmitted, meaning it contains the last decimated result from the filter.

37.3.2.7 Decimation Filter Coefficient n register (DECFILTER_COEFn)

Address: DECFILTER_BASE + 0x020–0x040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	8{COEFn[23]}								COEFn[23:16]							
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	COEFn[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 377. Decimation Filter Coefficient n register (DECFILTER_COEFn)

Table 447. DECFILTER_COEFn field descriptions

Field	Description
8-31 COEFn[23:0]	Coefficient n field The COEFn[23:0] bit fields are the digital filter coefficients registers. The coefficients are fractional signed values in two's complement format, in the range (-1 ≤ coef < 1). **Note:** Reads to this register are sign-extended, meaning the coefficient's sign bit is copied to all 8 most significant register bits. **Note:** Writing to these fields when BSY=1 is not allowed.

37.3.2.8 Decimation Filter TAPn register (DECFILTER_TAPn)

Address: DECFILTER_BASE + 0x078–0x094 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	8{TAPn[23]}								TAPn[23:16]							
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	TAPn[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 378. Decimation Filter TAPn Register (DECFILTER_TAPn)

Table 448. DECFILTER_TAPn field descriptions

Field	Description
8-31 TAPn[23:0]	The read-only TAPn[23:0] bit fields shows the contents of the digital filter tap registers, as fractional signed values in two's complement format, in the range (-1 ≤ coef < 1). **Note:** Reads to this register are sign-extended, meaning the coefficient's sign bit is copied to all 8 most significant register bits. **Note:** The content of these registers is meaningless when BSY=1.

37.3.2.9 Decimation Filter Final Integration Value register (DECFILTER_FINTVAL)

Address: DECFILTER_BASE + 0x0E0 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SUM_VALUE[31:16]															
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	SUM_VALUE[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 379. Decimation Filter Final Integration Value register (DECFILTER_FINTVAL)

Table 449. DECFILTER_FINTVAL field descriptions

Field	Description
0-31 SUM_VALUE[31:0]	Integration Sum Value The SUM_VALUE[31:0] field holds the sum of filtered output values. The 17 most significant bits hold the integer part, and the 15 least significant ones the fractional part of the integration value. The control of the integration sum and update of this register is determined by the register DECFILTER_MXCR (see *Section 37.3.2.3: Decimation Filter Module Extended Configuration Register (DECFILTER_MXCR)*). The register is updated only upon an integration output request. SUM_VALUE should be taken as an unsigned number when the integrator is configured for absolute operation (DECFILTER_MXCR bit SSIG=0), and a two's complement signed number otherwise. **Note:** If the DEFILTER_MXCR bit SSAT=1, the integration sum is saturated, so that if the accumulation overflows DECFILTER_FINTVAL holds the value 0xFFFFFFFF for absolute integration (SSIG=0), or values 0x7FFFFFFF (positive saturation) and 0x80000000 (negative saturation) for signaled integration (SSIG=1). **Note:** If SSAT=0, DECFILTER_FINTVAL holds the integration sum modulo 2^{17} (considering the 15-bit fractional part).

37.3.2.10 Decimation Filter Final Integration Count Value register (DECFILTER_FINTCNT)

Address: DECFILTER_BASE + 0x0E4 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	COUNT[31:16]															
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	COUNT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 380. Decimation Filter Final Integration Count Value register (DECFILTER_FINTCNT)

Table 450. DECFILTER_FINTCNT field descriptions

Field	Description
0-31 COUNT[31:0]	Integration Count Value The COUNT field holds the count of filtered outputs integrated. The control of the integration sum and update of this register is determined by the register DECFILTER_MXCR (see *Section 37.3.2.3: Decimation Filter Module Extended Configuration Register (DECFILTER_MXCR)*). The register is updated together with DECFILTER_FINTVAL, only upon an integration output request.

37.3.2.11 Decimation Filter Current Integration Value register (DECFILTER_CINTVAL)

Address: DECFILTER_BASE + 0x0E8 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SUM_VALUE[31:16]															
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	SUM_VALUE[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 381. Decimation Filter Current Integration Value Register (DECFILTER_CINTVAL)

Table 451. DECFILTER_CINTVAL field descriptions

Field	Description
0-31 SUM_VALUE[31:0]	Integration Sum Value The SUM_VALUE[31:0] field holds an unsigned number representing the sum of filtered output values, continuously updated as the integration proceeds. The control of the integration sum is determined by the register DECFILTER_MXCR (see *Section 37.3.2.3: Decimation Filter Module Extended Configuration Register (DECFILTER_MXCR)*). **Note:** If the DEFILTER_MXCR bit SSAT=1, the integration sum is saturated, so that if the accumulation overflows DECFILTER_CINTVAL holds the value 0xFFFFFFFF for absolute integration (SSIG=0), or values 0x7FFFFFFF (positive saturation) and 0x80000000 (negative saturation) for signaled integration (SSIG=1). If SSAT=0, DECFILTER_FINTVAL holds the integration sum modulo 2^{17} (considering the 15-bit fractional part). **Note:** A read on this register automatically commands an update of the register DECFILTER_CINTCNT.

37.3.2.12 Decimation Filter Current Integration Count Value register (DECFILTER_CINTCNT)

Address: DECFILTER_BASE + 0x0EC Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	COUNT[31:16]															
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	COUNT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 382. Decimation Filter Current Integration Count Value register (DECFILTER_CINTCNT)

Table 452. DECFILTER_CINTCNT field descriptions

Field	Description
0-31 COUNT[31:0]	Integration Count Value The COUNT field holds the count of filtered outputs integrated. The value is updated only when register DECFILTER_CINTVAL is read, to keep the coherency between the integration and count values.

37.4 Functional description

37.4.1 Overview

Figure 370 shows the block diagram of the Decimation Filter. The control logic provides the control signals for all other sub-modules.The data interface is sub-divided into two sub-modules, transmitter and receiver, that are accessed by the slave-bus interface. The bypass

path is used when the filter is disabled and the incoming data can be transmitted back to the master block without being processed by the Filter algorithm. The Filter hardware is implemented in such a way that an IIR (1 x 4 poles) or FIR filter type can be implemented. The selection between the two types of filter algorithms is implemented by the Control Logic sub-block.

The Coefficient register file provides the digital filter coefficients. This block is a register bank with read/write access by the device slave-bus interface. The Filter TAP registers are also accessed through the device slave-bus line interface, providing additional debug capabilities to the Decimation Filter block. The MAC (Multiply Accumulate) sub-block executes the filter arithmetic operations controlled by the Control Logic. The MAC results are routed to the Filter TAP registers and to the output buffer when the result is a decimated filter sample.

37.4.2 Data interface sub-block description

This section describes the operation of the data interface sub-block which is responsible for communication and data exchange between the slave-bus and the Decimation Filter block. It is important to note that there are only one internal input buffer and only one internal output buffer.

The decimation filter receives sample data from the memory . The input data bus format is presented in *Figure 375*. The sample data arrives along with the control bits . These control bits are decoded and the proper action is decided in the Control Logic sub-block. When the decimation filter finishes its processing and a result is available, the read request signal is issued to the selected master block. This data transfer request remains set until the result is read by the master block.

37.4.3 Input buffer description

The decimation filter receives data samples for filtering from the CPU using the device slave-bus interface. The data source is selected by the ISEL bit (ISEL=1) of the module configuration register DECFILTER_MCR.

When the device slave-bus interface is selected and DMA operation is chosen (DECFILTER_MCR bit DSEL=1), the input data request signal is asserted when the input buffer is empty. When DMA operation is not chosen (DSEL=0) in standalone mode, the logic asserts an input interrupt request and the input buffer waits for data from the device slave-bus.

Input buffer filling is flagged by the DECFILTER_MSR bit IDF. The IDF flag remains set, even after the input data has been consumed and the buffer is free, until it is cleared by software.

Input buffer overrun is detected and flagged by the DECFILTER_MSR bit IVR. The idle (BSY=0) Decimation Filter is able to receive two consecutive input writes without input overrun. For more details, see *Section 37.4.3.1: Input buffer overrun*.

Soft reset clears the input buffer indication flags and any data write/read request that was generated in any mode.

37.4.3.1 Input buffer overrun

An input overrun occurs when the input buffer is holding input data and new data is received by the filter. See *Section 37.4.3: Input buffer description*, for details of the input buffer.

The input buffer overrun can occur only when the input is enabled (DECFILTER_MCR bit IDIS=0).

37.4.4 Output buffer description

The decimation filter has an output buffer to send filtering results to the CPU using the device slave-bus, as selected through the DECFILTER_MCR bits ISEL. Filtering of prefill inputs do not update the output buffer, so the flag ODF is not set.

When in filter operation mode with input from the device slave-bus (ISEL=1), only sample data is processed by the filter. When the filter result from the MAC is ready, this is immediately written to the output buffer when the decimation count is reached. The flag ODF is set when the buffer is updated. It also generates a DMA read request if DSEL=1 in standalone mode.

When the filter is bypassed (FTYPE=00), and the device slave-bus is selected as output (standalone mode), the data written into the input buffer is immediately written into the output buffer. The flag ODF is set when the buffer is updated and a DMA read request is asserted if DSEL=1.

Soft reset clears the output buffer, as well as any data read request generated in any mode.

37.4.4.1 Output buffer overrun

An output overrun occurs when the output buffer is holding output data that has not been read is overwritten with another word of data.

Output overruns are flagged by the DECFILTER_MSR bit OVR, if the output buffer is updated when not empty. The output buffer empty condition depends on the mode and output selection as follows:

- if the device slave-bus is selected (standalone mode) for output and DMA is selected (DSEL=1), the output buffer is considered empty if the last output has been read.
- if the device slave-bus is selected (standalone mode) for output and DMA is not selected (DSEL=0), the output buffer is considered empty when the ODF flag is negated.

Note: When the device slave-bus is selected (standalone mode) for output and DMA is not selected (DSEL=0), the ODF flag must be cleared to avoid overrun, even if its corresponding interrupt is not used (ODEN=0).

Prefill inputs do not cause IIR or FIR output overrun in standalone mode.

When bypass is selected, the output overrun does not occur because the data written into the input buffer is written into the output buffer only when this buffer is empty, but an input overrun may still occur (see *Section 37.4.3.1: Input buffer overrun*).

37.4.5 Bypass configuration description

Bypass operation is configured by setting the field FTYPE[1:0] of module configuration register DECFILTER_MCR to 00. In this case, the input sample and tag are sent to the

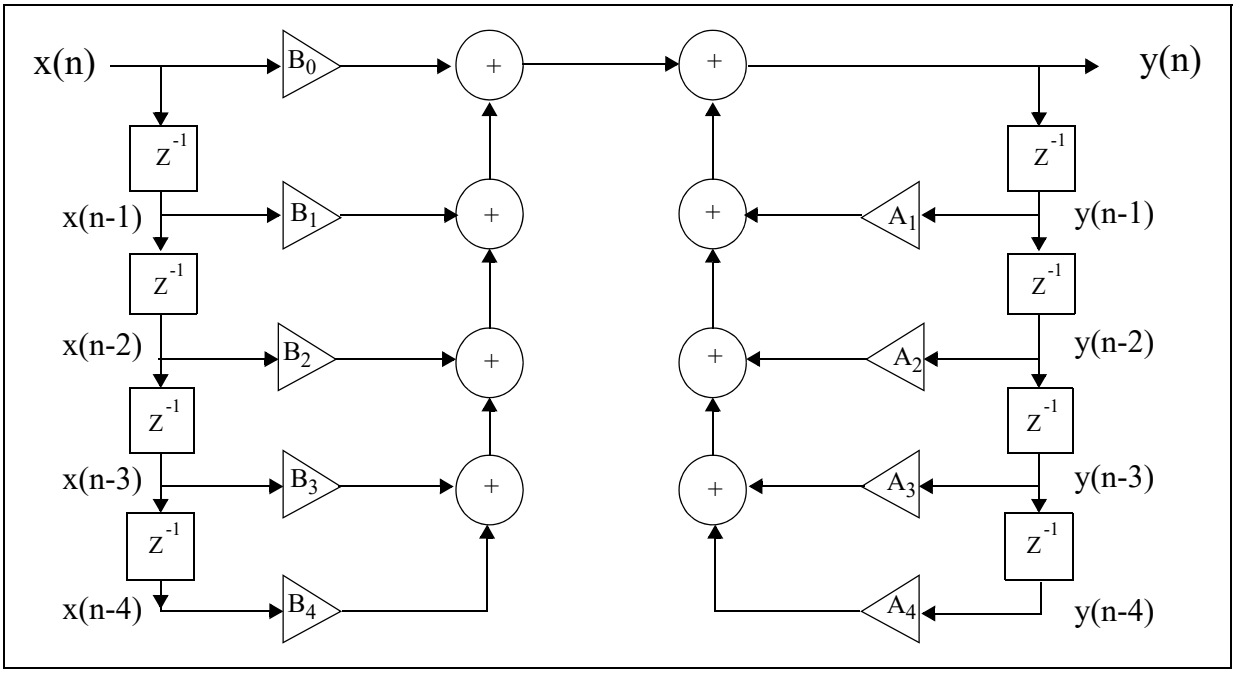
output with no change. This behavior is independent of the ISEL setting. The following applies to the bypass configuration:

- flush is ignored
- prefill is ignored
- counted decimation is ignored
- BSY bit is not set
- the input and output flags are set

37.4.6 IIR and FIR Filter

This section describes the IIR filters implemented in the Decimation Filter block. *Figure 383* shows the filters functional diagrams. The figure shows a IIR filter of fourth order (4poles). The filter topology, not the hardware, is represented. The hardware implementation is based on a MAC unit controlled by an FSM which implements the filter algorithm.

Figure 383. 1 x 4 Poles IIR Filter Functional Diagram



The difference equation for the IIR filter of *Figure 383* can be written as:

Equation 18

$$y(n) = \sum_{i=0}^{N} B_i x(n-i) + \sum_{j=1}^{M} A_j y(n-j)$$

where *x(n)* is the filter input at time *n, y(n)* is the filter output at time *n*, *N* is the number of feed-forward filter coefficients minus one, B_i are the feed-forward filter coefficients, *M* is the number of feed-back filter coefficients, and A_j are the feedback filter coefficients.

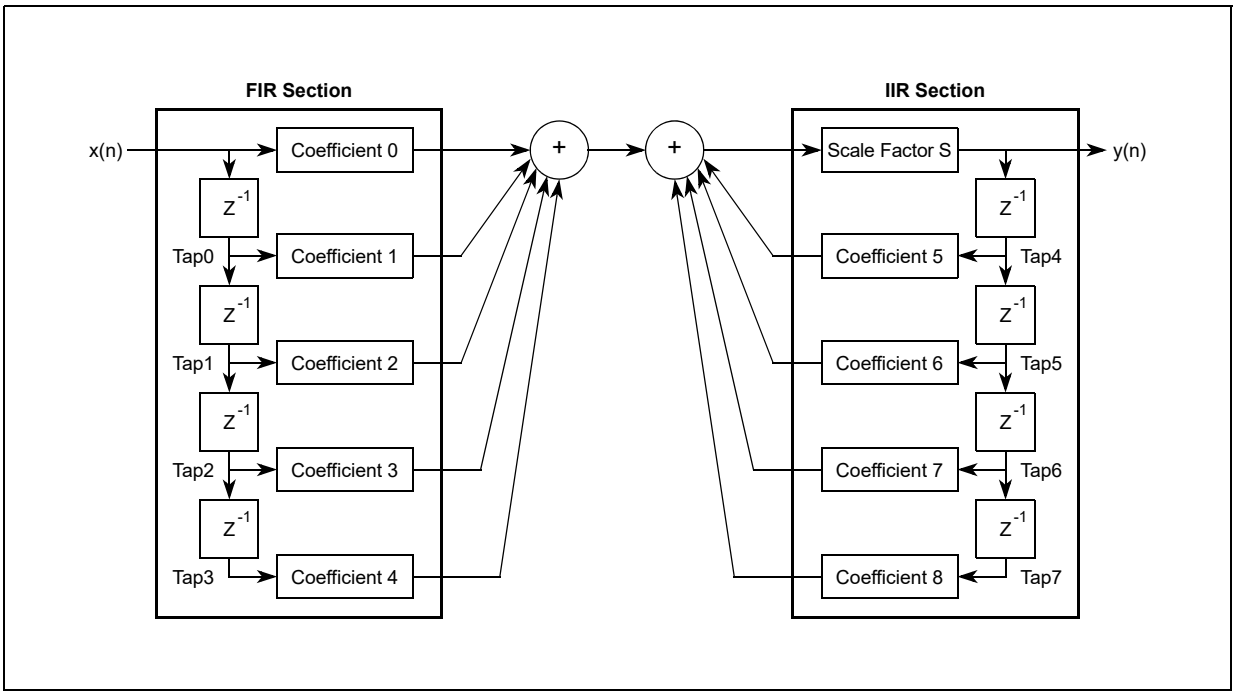
Equation 18 can be written as:

Equation 19

$$y(n) = S\left\{\sum_{i=0}^{N}\frac{B_i}{S}x(n-i) + \sum_{j=1}^{M}\frac{A_j}{S}y(n-j)\right\}$$

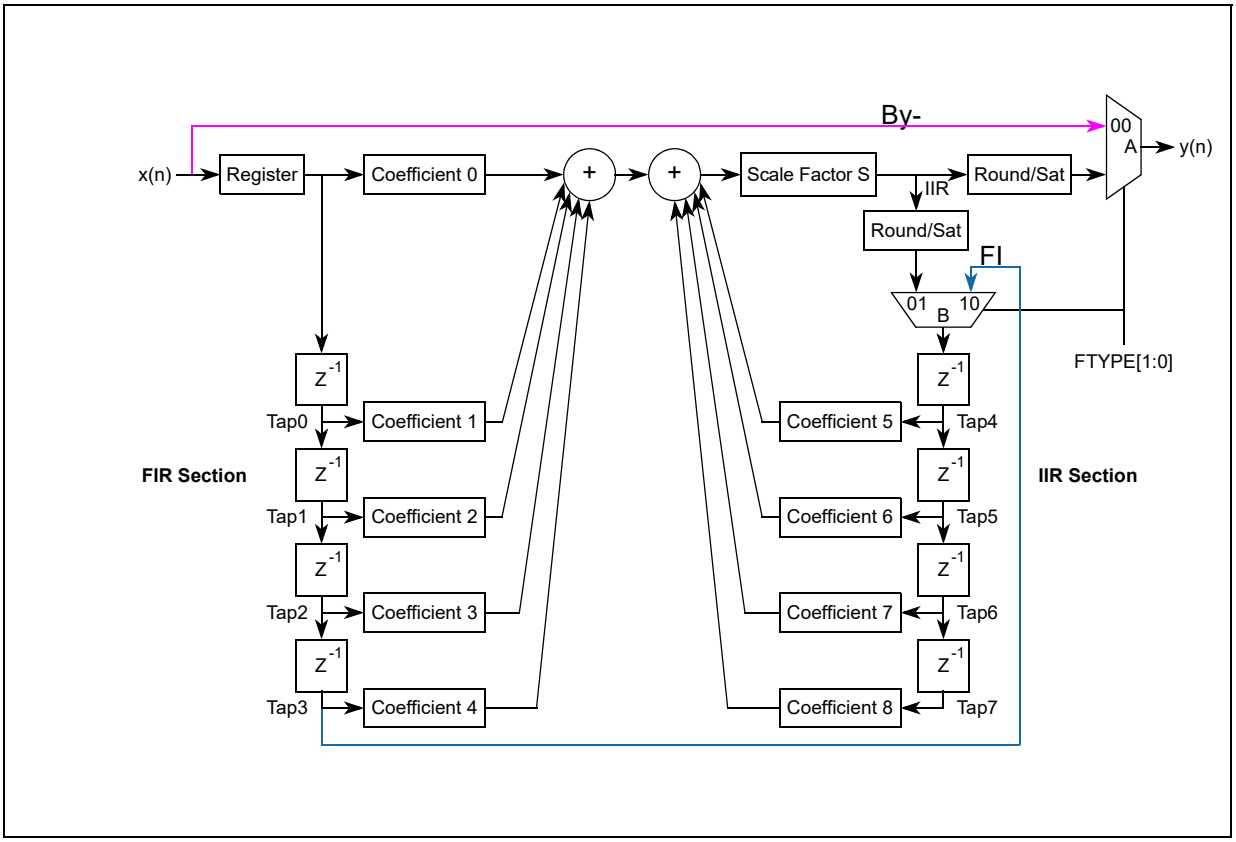
Where all the coefficients are scaled down by S. The block diagram for *Equation 19* is shown in *Figure 384* in a fourth-order IIR filter implementation where the coefficients A_j and B_i are called *coefficient n*, where *n*=0-8.

Figure 384. Fourth Order IIR Filter Implementation Block Diagram



The fourth order IIR filter is implemented with a FIR section followed by an IIR section. If the FIR type filter mode is selected, the IIR section is converted into an FIR section. In this case the order of the FIR filter is twice the IIR filter order, since all the TAP and coefficient registers are allocated for the FIR section. The Filter configuration paths are shown in *Figure 385*. Multiplexer A controls the *bypass* filter path and multiplexer B controls/selects the filter mode of operation, to either IIR mode or FIR mode. The selection is controlled by the FTYPE[1:0] bits in the Filter Module Configuration register. The order of the filter can be controlled by setting the appropriate *filter coefficients* to *zero.*

Figure 385. Filter Configuration Paths (FIR or 1x4Poles IIR)



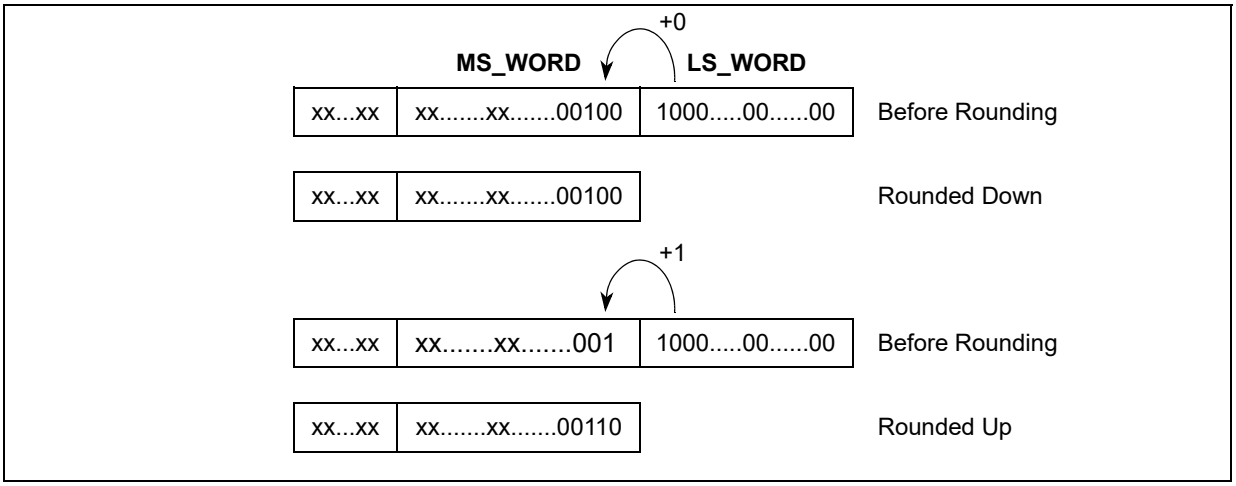
37.4.6.1 Rounding

The Decimation Filter performs rounding operations in two different locations, as shown in *Figure 385*:

- to obtain the filter output result with 16 bits
- to obtain the IIR feedback result to be stored in tap4 registers with 24 bits

The rounding mechanism implements the Convergent Rounding methodology (also known as round-to-nearest even number), which makes the decision on rounding up or down based on the value of the lower portion of data to be rounded (LS_WORD). The rounding up/down condition is equal to the traditional rounding except when the LS_WORD has the format {1000...00}. In this particular case, the rounding procedure is like the example of *Figure 386*. If the MS_WORD is odd, the value is rounded up. Otherwise the value is rounded down.

Figure 386. Convergent Rounding Methodology



37.4.6.2 Saturation

Filter output saturation occurs when an overflow or underflow condition of the filter is detected by dedicated logic, and if it is enabled by the SAT control bit of the configuration register DECFILTER_MCR. In this condition, the filter output is set to a saturated value equal to the maximum or minimum value that can be represented by the 16-bit output port. Also, for the IIR filter an equivalent logic is used to assert the saturation for the 24-bit feedback result.

37.4.7 Filter prefill control description

A prefill indicates that the input data should be accepted by the Decimation Filter, but no decimated output should be generated while the control field indicates prefill. Therefore the prefill function is used in the beginning of the filter operation to initialize and stabilize the Decimation Filter without generating decimated samples. In addition, the prefill does not operate when the filter is in bypass (FTYPE=0b00).

The prefill is controlled by the value in the M_CTRL[1:0] field in the DECFILTER_IOB register. When ISEL=1, it is controlled by the PREFILL field in the DECFILTER_IB register. The prefill control is usually activated only in a certain number of words of sample data in the beginning of the input data sequence.

When the prefill control is set, the decimation filter block operates as follows:

- Input data is processed normally by the digital filter and tap values are updated.
- The decimation counter is maintained in reset value.
- The output buffer is not updated and no output interrupt or read request is generated.
- The overflow detector/flag operates normally and the error interrupt request is set if enabled.

37.4.8 Flush command description

The flush signal is used by the Decimation Filter to execute a partial reset of the filter. This is useful when the same filter is used on a new set of data samples after finishing the filtering of another set of data.

When the flush control is detected, all filter TAPs are cleared and the DEC_COUNTER[3:0] field in the status register DECFILTER_MSR is reset.

The flush function does not clear the Coefficient registers file in the Decimation Filter, thus it is not required to re-write these registers after a flush. The integrator accumulator and sample count are not cleared either. The output buffer also keeps the last result and may be retrieved until the next output is posted.

The flush control precedes the input data to be filtered. Therefore, the corresponding sample data is processed by the block after the flush. When ISEL=1, the field FLUSH in the DECFILTER_IB register is processed. Note that a word of valid sample data can be available at the same time the flush signal is asserted. In this case the flush is executed and the sample is processed after the flush.

When the filter is disabled by the FTYPE[1:0] control bit field, the flush command is not executed.

37.4.9 Soft reset command description

The soft reset command is requested through the self-negated bit SRES of the DECFILTER_MCR register and provides the CPU with the capability to initialize the Decimation Filter through the slave-bus interface. The procedure below must be performed for a software reset when the filter is active:

1. disable filter inputs, writing DECFIL_MCR bit IDIS = 1.
2. poll the register DECFIL_MSR until the bit BSY is 0.
3. repeat the step 2 polling; this is necessary to cover the case when a sample is left in the input buffer.
4. write DECFILTER_MCR bit SRES = 1.

After the software reset is issued, all internal Filter TAP registers, the decimation counter, the integrator outputs (except DECFILTER_CINTCNT) and the state machine are put in the initial state. The status register DECFILTER_MSR is also cleared. The Coefficient registers are not affected by the SRES. In case there is some filter processing, the filter process is aborted and the last sample is discarded. In addition, data in the input buffer waiting to be processed, and data in the output buffer waiting to be read, are discarded (the requests of service are cleared). The software reset command has high priority and the BSY bit is set during its operation.

The configuration registers DECFILTER_MXCR and DECFILTER_MCR are also not affected by a soft reset, except the bit SRES that is self-negated and is always read as zero.

When in debug or freeze mode, the soft reset is executed but the filter remains in debug or freeze mode.

Note: It is recommended to clear the IBIE bit before a software reset, especially if ISEL changes, in order to avoid unwanted interrupt requests.

Note: DMA transfers must not be active during soft reset. Data loss can occur.

37.4.10 Interrupts requests description

37.4.10.1 Block interrupt request

There are several interrupt request events that can be enabled using the module configuration register DECFILTER_MCR. Basically, the interrupt request can be issued under any of the following conditions:

- when a word of input data is received
- when a word of output data is available
- when an error has occurred.

The input data flag IDF is set when a word of data is received from the CPU when in standalone mode. It is not used to generate the read or write requests (as defined in *Section 37.4.10.2: Input buffer interrupt request*) when DSEL=0.

Output data is available and its flag (ODF) is set when the input data sample is processed by the filter and the decimation counter matches the decimation rate value. It is not used to generate the read requests (as defined in *Section 37.4.10.3: Output buffer interrupt request*) when DSEL=0.

An error event in the decimation filter block is defined as one of these events:

- Overflow in the filter, flagged by OVF
- Overrun in the decimation filter input, flagged by IVR
- Overrun in the decimation filter output, flagged by OVR
- Integrator overrun, flagged by SVR
- Integrator value exception, flagged by SSE
- Integrator count exception, flagged by SCE

An overflow occurs when the two's-complement result value from the MAC accumulator is out of the range of values that can be stored in tap register 4 (IIR) or in the output register.

An input overrun occurs when the input buffer is holding a word of input data and one more word of data is received by the filter. See *Section 37.4.3.1: Input buffer overrun*, for more details.

An output overrun occurs when a new word of data is sent to the output buffer but the previous word of data has not been handled yet. See *Section 37.4.4.1: Output buffer overrun*, for more details.

These flags are only cleared by:

- the CPU, or
- by the soft reset command in the DEC_FILTER_MCR, or
- by the clear flag registers in the DEC_FILTER_MSR register.

37.4.10.2 Input buffer interrupt request

This interrupt is enabled by the register IBIE in the DECFILTER_MCR register and is asserted only when DSEL=0. This request is flagged in the DECFILTER_MSR by field IBIF.

In standalone mode, the input buffer interrupt request is asserted when the input buffer is available to receive a conversion sample and DMA operation is not selected (DSEL=0), meaning the block is requesting data to be written into the input buffer. The interrupt request is cleared when the CPU writes a one in field IBIC of the DECFILTER_MSR register, or by the soft reset command.

37.4.10.3 Output buffer interrupt request

This interrupt is enabled by the register OBIE in the DECFILTER_MCR register and is asserted only when DSEL=0. This request is also indicated in the field OBIF of the DECFILTER_MSR register.

When in standalone mode, the output buffer can be read by the CPU with the DMA disabled. The output buffer interrupt request is asserted when the output buffer receives a new result from the filter sub-block. This means the block is requesting data to be read by the CPU.

The output buffer interrupt request can also be asserted due to an integrator result being ready, as flagged by the DECFILTER_MXSR bit SDF, when the DECFILTER_MCR bit SDIE=1. Note that both the filter output and integrator share the same interrupt source.

This interrupt request is cleared when a one is written in the bit OBIC and/or bit SDFC of the DECFILTER_MXSR register, or by the soft reset command.

37.4.11 DMA requests description

The DMA function for input and output buffers is enabled using DSEL=1 in the DECFILTER_MCR register. The DMA function for integrator result is described in another section.

37.4.11.1 Input buffer DMA request

This DMA request is enabled by the ISEL and DSEL bits in the DECFILTER_MCR register.

When in standalone mode, the input buffer can be written by the DMA. The input buffer DMA request is asserted when the input buffer is available to receive a conversion sample (it is not holding a word of data). This DMA request is cleared when an input data word is written in the input buffer. Therefore, the DMA request is always cleared before it is asserted again.

This DMA request is cleared when the DMA transfer is complete or by soft reset.

37.4.11.2 Output buffer DMA request

This DMA request is enabled by the ISEL and DSEL bits in the DECFILTER_MCR register.

When in standalone mode, the output buffer can be read using the DMA. The output buffer DMA request is asserted when the output buffer receives a new result from the internal filter sub-block. This DMA request is cleared when the output buffer is read by the processor.

The output buffer DMA request is also asserted when an integrator output is ready, if DECFILTER_MXCR bit SDMAE=1 with DSEL=0.

The DMA request is also cleared by soft reset.

37.4.12 Freeze mode description

The freeze mode operation is asserted using the FREN enable bit and FRZ bit in the DEC_FILTER_MCR register or by the modules debug input.

It is not possible to enter freeze mode when the module is disabled by the configuration bit MDIS.

In case of a freeze mode request during the processing of an input sample, the current processing is finished and then the module enters freeze mode.

Access to input and output buffers remain operational in freeze, as well as their related flags.

37.4.13 Integrator

The hardware sample integrator accumulates the filter output values for determined periods.

37.4.13.1 Integrator inputs

The integrator can be fed either by raw or decimated filter outputs, selected by the DECFILTER_MXCR bit SISEL (see *Section 37.3.2.3: Decimation Filter Module Extended Configuration Register (DECFILTER_MXCR)*). The accumulated input value taken can be the filtered sample “as is” (signaled), or its absolute value, depending on the DECFILTER_MXCR register bit SSIG.

Note: The integrator accumulates input samples when bypass is selected.

37.4.13.2 Integrator outputs

The integrator output is either:

- the 32-bit, fixed point unsigned accumulation of the absolute values from the filter output, when the integrator is configured for absolute operation (DECFILTER_MXCR bit SSIG=0). This resolution allows a minimum of 131071 samples to be integrated before an overflow occurs in absolute operation, or 65536 samples in signed operation.
- the 32-bit, fixed point signed two’s complement accumulation of the signed values from the filter output, when the integrator is configured for signed operation (SSIG=1).

The fractional part of the accumulation is 15 bits wide in both cases.

An accumulation overflow is flagged by the DECFILTER_MXSR bit SSOVF. The accumulator can overflow in either of the ways below, selected through the DECFILTER_MXCR bit SSAT:

- saturated accumulation (SSAT=1), so that an overflow results in the value of 0xFFFFFFFF for absolute value accumulation (SSIG=0), or 0x7FFFFFFF (positive) and 0x80000000 (negative) for signaled accumulation (SSIG=1).
- non-saturated accumulation (SSAT=0), so that an overflow results in the modulo 2^{17} accumulation value. This operation is only allowed in absolute accumulation (SSIG=0).

The integrator output value becomes available in register DECFILTER_FINTVAL (see *Section 37.3.2.9: Decimation Filter Final Integration Value register (DECFILTER_FINTVAL)*) when an integrator output request is issued. The integrator output request can be issued in the following ways:

- by hardware, controlled by an external signal ; the enabling and the selection of the signal request modes is done through the DECFILTER_MXCR field SRQSEL (see *Section 37.3.2.3: Decimation Filter Module Extended Configuration Register (DECFILTER_MXCR)*);
- by software, writing 1 to the DECFILTER_MXCR bit SRQ;

The SSOVF flag is asserted upon an integrator output request, based on the overflow state of the internal accumulator. This internal overflow state is cleared upon an output request, just after SSOVF is asserted, or upon an integrator reset (see *Section 37.4.13.3: Integrator reset*). The internal overflow state is also cleared by writing SSOVFC to 1, but only in saturated accumulation. Therefore, a non-saturated overflow that occurs before an SSOVF clear is still flagged in the next output request.

The integrator output request also updates the register DECFILTER_FINTCNT, which holds the number of samples accumulated into the register DECFILTER_FINTVAL. This internal accumulated sample counter can operate either in a saturated or “wrapped” count mode, as selected by the DECFILTER_MXCR bit SCSAT. In both cases, the counter overflow is flagged by the DECFILTER_MXSR bit SCOVF.

The SCOVF flag is asserted upon an integrator output request, based on the overflow state of the internal counter. This internal overflow state is negated upon an output request, just after SCOVF is asserted, or upon an integrator reset (see *Section 37.4.13.3: Integrator reset*”). The internal overflow state is also negated by writing SCOVFC to 1, but only in saturated count. Therefore, a non-saturated overflow that occurs before an SCOVF clear is still flagged in the next output request.

An integrator output update can also issue a DMA or interrupt request. The interrupt and DMA requests are the same ones used for the filter output buffer (see *Section 37.4.10.3: Output buffer interrupt request* and *Section 37.4.11.2: Output buffer DMA request*). The DECFILTER_MCR bit SDIE is used to enable integrator interrupts, and the DECFILTER_MXCR bit SDMAE enables the DMA integrator requests. The integrator DMA request uses the same signal as the filter output DMA request, so one must never use any configuration that allows both the integrator and filter output to make DMA requests.

Integrator output updates are flagged by the DECFILTER_MXSR bit SDF. The integrator overrun is detected in the same way as a filter output buffer overrun, and is flagged by DECFILTER_MXSR bit SVR. An integrator overrun also generates an error interrupt if the DECFILTER_MCR bit ERREN=1 (see *Section 37.4.4.1: Output buffer overrun*).

Registers DECFILTER_CINTVAL and DECFILTER_CINTCNT provide a way to poll intermediate integration values and sample counts, respectively (see *Section 37.3.2.11: Decimation Filter Current Integration Value register (DECFILTER_CINTVAL)* and *Section 37.3.2.12: Decimation Filter Current Integration Count Value register (DECFILTER_CINTCNT)*). DECFILTER_CINTVAL is updated whenever the integrator is reset or a new sample is accumulated. DECFILTER_CINTCNT is updated only when DECFILTER_CINTVAL is read, so that coherency between the value and count values is guaranteed. Therefore, the read access order of that pair of registers must be DECFILTER_CINTVAL first, followed by DECFILTER_CINTCNT.

Note: The flags SSOVF and SCOVF can also asserted when DECFILTER_CINTVAL is read. The SSOVF and SCOVF set and clearing rules apply for the DECFILTER_CINTVAL read the same way as for an integrator output request.

37.4.13.3 Integrator reset

The integration value is reset to the value of zero, in the following ways:

- by hardware: on hardware reset, or controlled by an external signal ; the enabling and the selection of the zero signal modes is done through the DECFILTER_MXCR field SZROSEL (see *Section 37.3.2.3: Decimation Filter Module Extended Configuration Register (DECFILTER_MXCR)*);
- by software: on software reset, or writing 1 to the DECFILTER_MXCR bit SZRO;

The integrator reset also zeroes the internal counter of accumulated samples and the internal overflow state (but not SSOVF and SCOVF). Software and hardware resets reset all

integrator registers immediately. Integrator zero command from external signal or by software (SZRO) affects the integrator registers and flags as follows:

- DECFILTER_CINTVAL resets immediately;
- DECFILTER_CINTCNT does not reset immediately; it is updated only upon a DECFILTER_CINTVAL read, loaded with the number of integrated samples occurred after the reset;
- DECFILTER_FINTVAL and DECFILTER_FINTCNT do not reset immediately; being updated only upon a new output request (see *Section 37.4.13.2: Integrator outputs*); if a integrator software zero command (through SZRO bit) and an integrator output request (through SRQ bit) are made at the same time, the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT are updated with the last internal values before reset; the same applies to simultaneous integrator zero command and output request by hardware signal;
- the SSOVF and SCOVF flags do not negate; however, the internal overflow states which assert SSOVF and SCOVF do reset immediately, so that the next output update (either by hardware request, software request or DECFILTER_CINTVAL read) before an overflow does not assert SSOVF/SCOVF.

Note: The integrator reset does not depend on the integrator enabling (see Section 37.4.13.4: Integrator enabling and halting).

37.4.13.4 Integrator enabling and halting

Two mechanisms, enabling and halting, drive the integrator accumulation, allowing it to be controlled by a combination of two distinct sources: both software, both hardware, or one hardware and the other software. Values are accumulated when the integrator is enabled and not halted. The integrator halt and enable states can be controlled in the following ways:

- by hardware, through external signals ; the enabling and the selection of the signal request modes is done through the DECFILTER_MXCR fields SENSEL and SHLTSEL, respectively (see *Section 37.3.2.3: Decimation Filter Module Extended Configuration Register (DECFILTER_MXCR)*);
- by software, through the same DECFILTER_MXCR fields SENSEL and SHLTSEL. Note that these fields are in different bytes, so that two distinct, concurrent software tasks can avoid coherency problems by changing the fields using byte read-modify-write accesses.

Note: Enabling and halting do not affect output requests or integrator reset.

37.4.13.5 Integrator exceptions

Integrator may run into exception states due to the overflow of either the accumulated value or the sample counter. Exceptions are flagged by the DECFILTER_MXSR bits SSE for sum value exception, and SCE for counter exception. These flags generate an error interrupt if it is enabled (see *Section 37.4.10.1: Block interrupt request*).

The accumulator exception condition depends on whether or not it operates in saturated mode as follows:

- In Saturated operation (DECFILTER_MXCR bit SSAT=1): a sum exception occurs (SSE=1) whenever an overflow is flagged; SSE asserts together with SSOVF.
- In Non-saturated operation (DECFILTER_MXCR bit SSAT=0): a sum exception occurs (SSE=1) when an overflow is flagged and the DECFILTER_MXSR bit SSOVF is already set to 1.
- In Non-saturated operation, an accumulator exception also occurs if the accumulator overflows twice without any update of the final integrator value DECFILTER_FINTVAL or the current integrator counter DECFILTER_CINTCNT (by a read to the DECFILTER_CINTVAL register), neither an integrator reset occurs. The SSOVF flag does not assert in this situation.

Note: The SSOVF flag can only be asserted upon a hardware request, a software request, or when DECFILTER_CINTVAL is read, based on the internal accumulator overflow state.

Similarly, the sample counter exception condition depends on whether or not it operates in saturated mode as follows:

- In Saturated operation (DECFILTER_MXCR bit SCSAT=1): a counter exception occurs (SCE=1) whenever an overflow is flagged; SCE asserts together with SCOVF.
- In Non-saturated operation (DECFILTER_MXCR bit SCSAT=0): a counter exception occurs (SCE=1) when an overflow is flagged and the DECFILTER_MXSR bit SCOVF is already set to 1.
- In Non-saturated operation, a counter exception also occurs if the counter overflows twice without any update of the final count DECFILTER_FINTCNT or the current integrator counter DECFILTER_CINTCNT (by a read to the DECFILTER_CINTVAL register), neither an integration reset occurs. The SCOVF flag does not assert in this situation.

Note: The SCOVF flag can only be asserted upon a hardware request, a software request, or when DECFILTER_CINTVAL is read (also updating DECFILTER_CINTCNT), based on the internal counter overflow state.

37.5 Initialization Information

Following are some simple initialization steps to be performed before using the decimation filter block. These steps assume that the user has already calculated the filter coefficients using a filter design tool.

37.5.1 Initialization Procedure

The sequence of steps for the block initialization is as follows:

1. Program the configuration registers DECFILTER_MCR and DECFILTER_MXCR as desired for your application.
2. Write all filter coefficient registers DECFILTER_COEFn with the previously calculated values.
3. Enable the filter input, writing DECFILTER_MCR bit IDIS=0.
4. Run a soft-reset cycle if necessary.
5. The module is ready to receive data from the device slave-bus interface.

37.6 Filter example simulation

The decimation filter block operation was checked in a Verilog simulation using calculated filter coefficient values and noisy input data. The expected output values and the RMS error were then calculated.

37.6.1 Coefficients calculation

The coefficients were calculated using a digital filter design tool. We have supplied some hypothetical filter parameters to the tool and obtained the filter coefficients. The input parameters are:

- filter characteristics: elliptic/low pass
- filter type: 4th order IIR
- input sample rate: 800k sample/s
- passband edge: 100 kHz
- stopband edge: 150 kHz
- passband attenuation: ≤ 1 dB

The software tool gives the IIR filter coefficients in the Z-transform format expressed by *Equation 20*:

Equation 20

$$\frac{Y(s)}{X(s)} = \frac{B_0 + B_1 s + B_2 s^2 + B_3 s^3 + B_4 s^4}{A_0 + A_1 s + A_2 s^2 + A_3 s^3 + A_4 s^4}$$

The coefficient results in fixed point decimal representation are shown in *Table 453*.

Table 453. Coefficient values given by SPW digital filter design tool

Coefficient	Decimal Value	Coefficient	Decimal Value
B0	0.0221455	A0	1.0
B1	0.00445582948893748	A1	-2.69772868375858
B2	0.0318517846509088	A2	3.234056294853
B3	0.00445582948893748	A3	-1.92028561712454
B4	0.0221455	A4	0.47939080709495

Comparing *Equation 19* with *Equation 20*, we obtain the relationship between the calculated values of coefficients and the values to be loaded in the DECFILTER_COEFn registers. Note that A0=1 in both cases and that the coefficients A1 to A4 in *Equation 20* has the opposite signal from *Equation 19*.

See *Table 454* below to obtain the coefficients. A scale factor of eight is used, being the smallest divider factor to have all coefficient values in the range (–1 ≤ Coef < +1).

Table 454. Coefficient values for decimation filter

SCALE	S=1	S=8	
COEFn	Decimal Value	Decimal Value	Hexadecimal Values (24 bits)
Coef0 = B0/S	0.0221455	0.00276815891266	0x005AB5
Coef1 = B1/S	0.00445582948893748	0.00055694580078	0x001240
Coef2 = B2/S	0.0318517846509088	0.00398147106171	0x008277
Coef3 = B3/S	0.00445582948893748	0.00055694580078	0x001240
Coef4 = B4/S	0.0221455	0.00276815891266	0x005AB5
Coef5 = –A1/S	2.69772868375858	0.33721613883972	0x2B29E6
Coef6 = –A2/S	–3.234056294853	–0.40425717830658	0xCC414E
Coef7 = –A3/S	1.92028561712454	0.24003565311432	0x1EB97D
Coef8 = –A4/S	–0.47939080709495	–0.05992400646210	0xF8546A

37.6.2 Input data calculation

The 24-bit words of input data are samples of a sum of two tones: one tone of 30 kHz and another tone of 250 kHz. The samples were calculated at the rate of 800k samples per second. The tones have the same amplitude and it is assured that the resultant amplitude is smaller than ‘1’ so as to obtain samples in the range ($-1 \leq$ sample $< +1$). It is supposed the input data are signed values in the two’s complement format in the range ($1 \leq$ sample $< +1$).

37.6.3 Filter results

The decimation filter block was used in a Verilog simulator using the calculated coefficients and the input data samples. A scaling factor of eight in the configuration register DECFILTER_MCR, and no decimation factor, were used to obtain the maximum of output results from the filter.

The theoretical expected values from this filter were also calculated, and these results were compared with those from the decimation filter. The resultant RMS error when considering about 500 samples was about –97 dB.

38 Temperature Sensor

38.1 Introduction

The device includes two on-chip temperature sensors that monitor device temperature and delivers one analog output signal and three digital output signals each. The analog output consists of a voltage signal directly proportional to the internal junction temperature. The analog output is connected to an input channel of an ADC on the device. The internal junction temperature must be calculated by software based on the sampled temperature sensor voltage, sampled reference voltage, which comes from another module, and calibration parameter values stored in internal flash memory. The three digital outputs, connected to the PMC module, are used to signal under- and over-temperature operating conditions. The digital outputs signal when the junction temperature drifts below the low temperature threshold or above one of two high temperature thresholds. These signals notify the device to take action to appropriately adjust the device temperature in response to an out of specification low or high temperature operating condition. Calibration parameter values associated with the temperature threshold detection feature are determined and stored in internal flash memory during production testing at the factory.

38.1.1 Signal Descriptions

This module has no external signals.

38.2 Functional Description

The temperature sensor generates one analog output voltage, VTSENS(T), which is proportional to the absolute current junction temperature of the device, and three digital outputs that signal whether the junction temperature has reached either a pre-set low temperature threshold or one of two pre-set high temperature thresholds.

An on-chip ADC module is used to convert the analog output voltage, as well as the bandgap voltage, into a digital representation. These values, along with parameter values stored in on-chip flash memory, are used by software to calculate the device junction temperature.

38.2.1 Temperature threshold detection (digital output generation)

Temperature threshold trimming values are adjusted through calibration during factory test and stored in on-chip flash memory. While enabled, if a new trimming value is loaded into the Temperature Sensor module, it may take up to 1 ms for the digital outputs to be updated to reflect the new trimming condition.

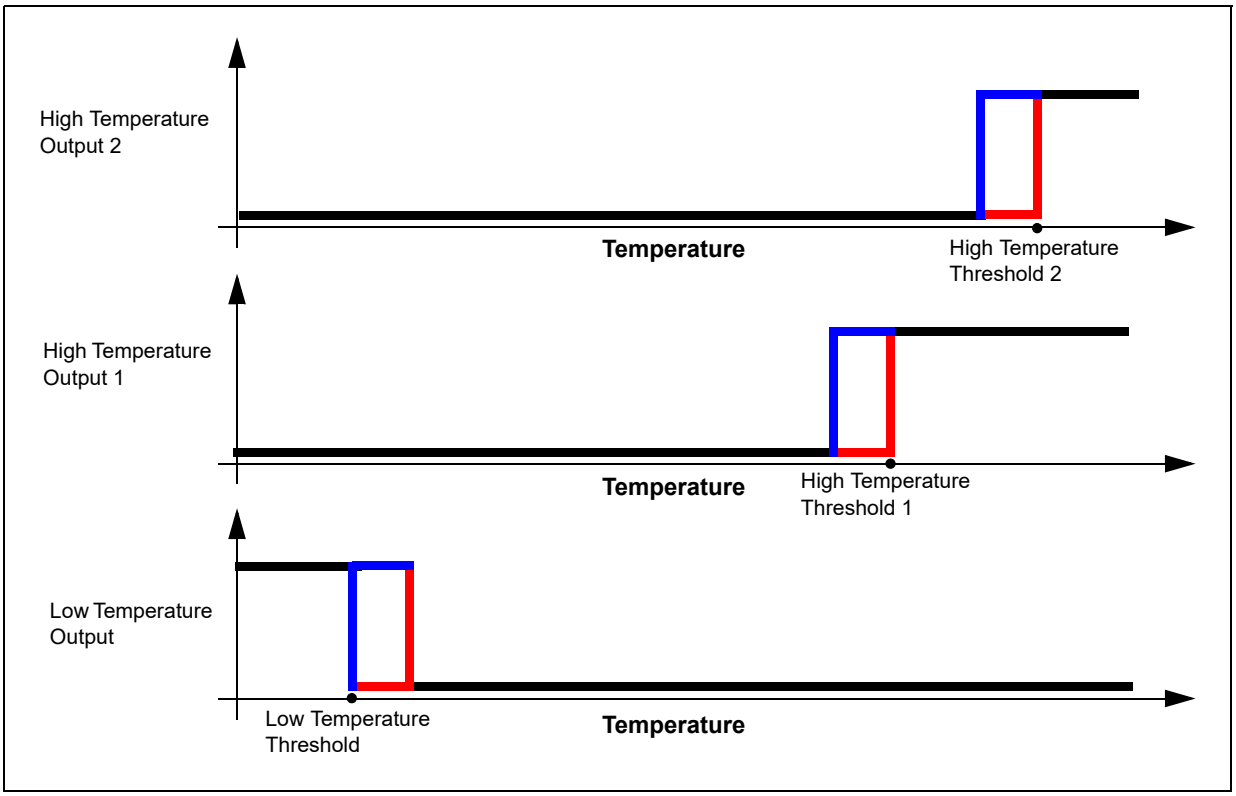
When the temperature threshold detection feature is enabled , the temperature sensor monitors the internal junction temperature of the chip and asserts a signal if any of the three specified temperature thresholds are crossed.

- The low temperature digital output signals if the junction temperature falls below the low temperature threshold (–40 °C).
- The high temperature digital output 1 signals if the junction temperature rises above the first high temperature threshold (150 °C).
- The high temperature digital output 2 signals if the junction temperature rises above the second high temperature threshold (165 °C).

The threshold detection circuit is calibrated with respect to the factory-test temperatures: the low temperature threshold is trimmed to match the cold insertion test temperature (nominally equal to –40 °C) and the high temperature thresholds are trimmed such that one of them matches the hot insertion test temperature (nominally equal to 150 °C or 165 °C). In other words, the thresholds are not trimmed to absolute values of temperature but to the specific temperatures for which the device is validated during factory test.

Hysteresis is applied whenever outputs transition to a logic level low in order to eliminate spurious toggling. Digital output behavior is illustrated on *Figure 387*.

Figure 387. Digital output behavior with temperature



38.2.2 Linear temperature sensor (analog output generation)

When analog output generation is enabled, the temperature sensor outputs a voltage proportional to the internal junction temperature of the chip. This analog voltage signal is converted into a digital code by an on-chip ADC. The temperature value is obtained from a

linear voltage-temperature relation with coefficients adjusted by calibration parameters extracted during factory test and programmed into flash memory.

38.3 Temperature formula

The system chain that translates device junction temperature into a digital variable is comprised of the temperature sensor, a reference voltage source and the on-chip ADC. Both analog output voltages of the temperature sensor and the reference voltage source must be converted by the ADC into digital codes to obtain the device junction temperature. The temperature formula shown in *Figure 388* is used to calculate internal device junction temperature.

Figure 388. Temperature formula

$$T = (232 + TSCA \times 2^{-6}) \times \frac{V_{TSENS_CODE}(T)}{V_{BG_CODE}(T)} - (273 + TSCB \times 2^{-4})\ [°C]$$

Where:

$$TSCA = \left(\frac{T_{HIGH} - T_{LOW}}{V_{TSENS_CODE}(T_{HIGH}) - V_{TSENS_CODE}(T_{LOW})} \times V_{BG_CODE}(T_{FINAL}) - 232 \right) \times 2^{6}$$

$$TSCB = \left(\frac{T_{HIGH} - T_{LOW}}{V_{TSENS_CODE}(T_{HIGH}) - V_{TSENS_CODE}(T_{LOW})} \times V_{TSENS_CODE}(T_{LOW}) - T_{LOW} - 273 \right) \times 2^{4}$$

$$V_{TSENS_CODE}(T_{LOW}) = \frac{V_{TSENS}(T_{LOW})}{V_{ref0}} \times 2^{12}$$

$$V_{TSENS_CODE}(T_{HIGH}) = \frac{V_{TSENS}(T_{HIGH})}{V_{ref0}} \times 2^{12}$$

$$V_{BG_CODE}(T_{FINAL}) = \frac{V_{BG}(T_{FINAL})}{V_{ref0}} \times 2^{12}$$

$$V_{BG_CODE}(T) = \frac{V_{BG}(T)}{V_{ref}} \times 2^{12}$$

$$V_{TSENS_CODE}(T) = \frac{V_{TSENS}(T)}{V_{ref}} \times 2^{12}$$

Notes:

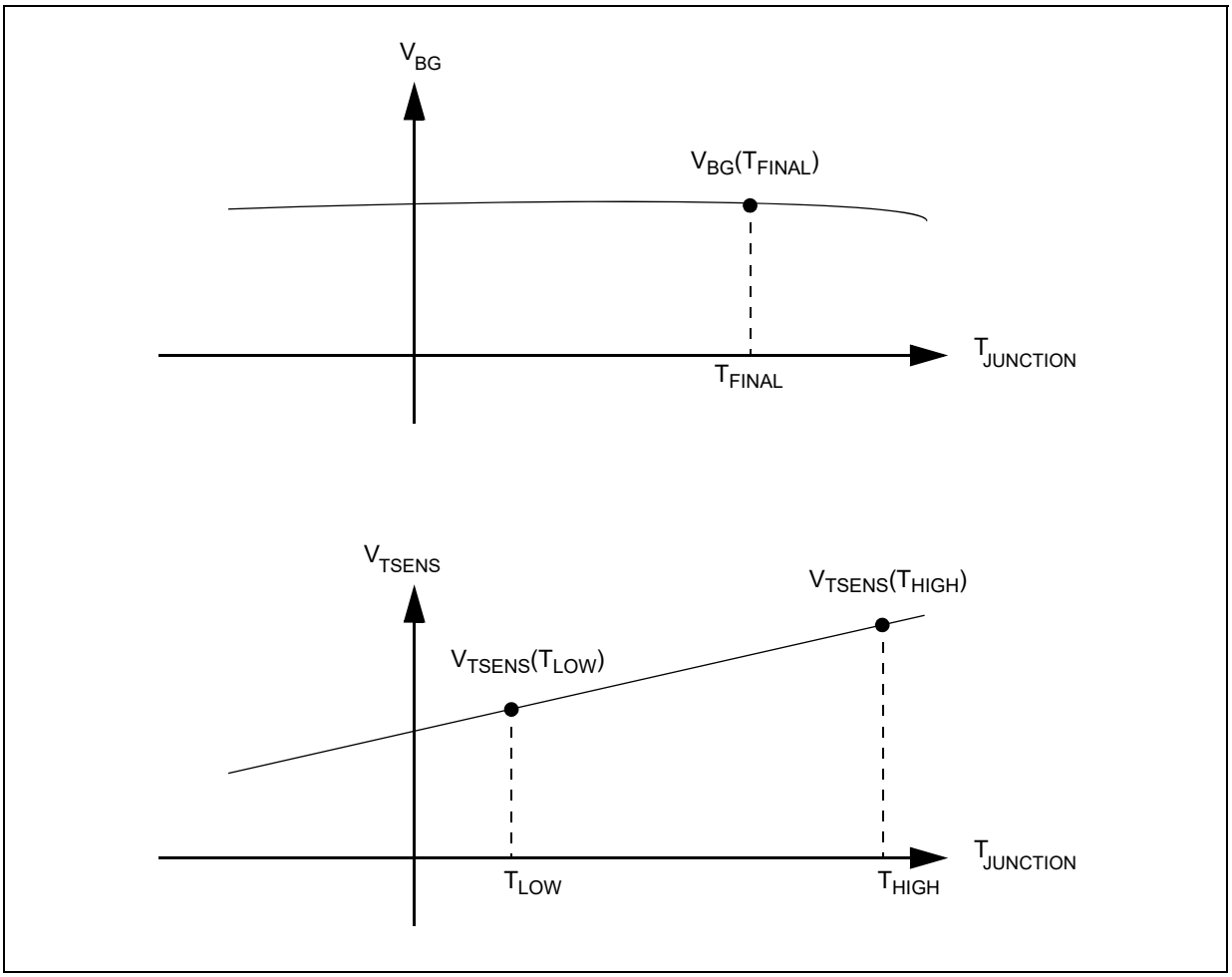
- TSCA and TSCB are 16-bit signed integer constants stored in flash memory during factory test and calibration
- $V_{TSENS}(T)$ is the temperature sensor output sampled by the ADC
- $V_{BG}(T)$ is the bandgap voltage sampled by the ADC at temperature T
- V_{ref} is the ADC reference voltage in normal usage of the device
- V_{ref0} is the ADC reference voltage during factory calibration
- T_{LOW} is the low temperature factory calibration temperature
- T_{HIGH} is the hot temperature factory calibration temperature
- T_{FINAL} is the final insertion factory calibration temperature

Parameters TSCA and TSCB are 16-bit signed integer constants stored during factory test and calibration that need to be used in the temperature formula to calculate device junction temperature. The values for TSCA and TSCB translate into positive or negative shifts of the linear relation coefficients that compensate for manufacturing process variation. *Figure 388*

shows how parameters TSCA and TSCB are calculated in terms of the actual variables measured during factory calibration.

The calibration points are illustrated in *Figure 389*.

Figure 389. Calibration Points



38.4 Calculating device temperature

Software must perform the following steps in order to calculate device temperature:

1. Measure the bandgap voltage using the on-chip ADC module and calculate $V_{BG_CODE}(T)$ according to the formula shown in *Figure 388*.
2. Measure the temperature sensor output voltage using the on-chip ADC module and calculate $T_{TSENS_CODE}(T)$ according to the formula shown in *Figure 388*
3. Retrieve parameters TSCA and TSCB from flash memory.
4. Calculate T according to the formula shown in *Figure 388*.

Note: Temperature formula may be calculated using signed 32-bit (integer) operations showing negligible precision loss.

39 System Timer Module (STM)

39.1 Introduction

39.1.1 Overview

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

39.1.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

39.1.3 Modes of operation

The STM supports two device modes of operation: normal and debug. When the STM is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the STM_CR register. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run.

39.2 External signal description

The STM does not have any external interface signals.

39.3 Memory map and register definition

The STM programming model has fourteen 32-bit registers. The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

39.3.1 Memory map

The STM memory map is shown in *Table 455*.

Table 455. STM memory map

Address offset	Register	Location
0x0000	STM Control Register (STM_CR)	*on page 810*
0x0004	STM Counter Value (STM_CNT)	*on page 811*
0x0008	Reserved	
0x000C	Reserved	

Table 455. STM memory map(Continued)

Address offset	Register	Location
0x0010	STM Channel 0 Control Register (STM_CCR0)	*on page 811*
0x0014	STM Channel 0 Interrupt Register (STM_CIR0)	*on page 812*
0x0018	STM Channel 0 Compare Register (STM_CMP0)	*on page 812*
0x001C	Reserved	
0x0020	STM Channel 1 Control Register (STM_CCR1)	*on page 811*
0x0024	STM Channel 1 Interrupt Register (STM_CIR1)	*on page 812*
0x0028	STM Channel 1 Compare Register (STM_CMP1)	*on page 812*
0x002C	Reserved	
0x0030	STM Channel 2 Control Register (STM_CCR2)	*on page 811*
0x0034	STM Channel 2 Interrupt Register (STM_CIR2)	*on page 812*
0x0038	STM Channel 2 Compare Register (STM_CMP2)	*on page 812*
0x003C	Reserved	
0x0040	STM Channel 3 Control Register (STM_CCR3)	*on page 811*
0x0044	STM Channel 3 Interrupt Register (STM_CIR3)	*on page 812*
0x0048	STM Channel 3 Compare Register (STM_CMP3)	*on page 812*
0x004C - 0x3FFF	Reserved	

39.3.2 Register descriptions

The following sections detail the individual registers within the STM programming model.

39.3.2.1 STM Control Register (STM_CR)

The STM Control Register (STM_CR) includes the prescale value, freeze control and timer enable bits.

Offset: 0x000 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CPS								0	0	0	0	0	0	FRZ	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 390. STM Control Register (STM_CR)

Table 456. 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. 0 STM counter continues to run in debug mode. 1 STM counter is stopped in debug mode.
31 TEN	Timer Counter Enabled. 0 Counter is disabled. 1 Counter is enabled.

39.3.2.2 STM Count Register (STM_CNT)

The STM Count Register (STM_CNT) holds the timer count value.

Offset: 0x004 Access: Read/Write

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

Figure 391. STM Count Register (STM_CNT)

Table 457. 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.

39.3.2.3 STM Channel Control Register (STM_CCR*n*)

The STM Channel Control Register (STM_CCR*n*) has the enable bit for channel n of the timer.

Offset: 0x10+0x10*n 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 392. STM Channel Control Register (STM_CCR*n*)

Table 458. STM_CCR*n* field descriptions

Field	Description
31 CEN	Channel Enable. 0 The channel is disabled. 1 The channel is enabled.

39.3.2.4 STM Channel Interrupt Register (STM_CIR*n*)

The STM Channel Interrupt Register (STM_CIR*n*) has the interrupt flag for channel n of the timer.

Offset: 0x14+0x10*n 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 393. STM Channel Interrupt Register (STM_CIR*n*)

Table 459. STM_CIR*n* field descriptions

Field	Description
31 CIF	Channel Interrupt Flag 0 No interrupt request. 1 Interrupt request due to a match on the channel.

39.3.2.5 STM Channel Compare Register (STM_CMP*n*)

The STM channel compare register (STM_CMP*n*) holds the compare value for channel n.

Offset: 0x18+0x10*n 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 394. STM Channel Compare Register (STM_CMP*n*)

Table 460. STM_CMP*n* field descriptions

Field	Description
0:31 CMP	Compare value for channel n. If the STM_CCR*n*[CEN] bit is set and the STM_CMP*n* register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIR*n*[CIF] bit is set.

39.4 Functional description

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel.

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the 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 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.

40 Software Watchdog Timer (SWT)

40.1 Introduction

40.1.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires the SWT generates an interrupt or hardware reset. The SWT can be configured to generate a reset or interrupt on an initial time-out. A reset is always generated on a second consecutive time-out.

40.1.2 Features

The SWT has the following features:

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

40.1.3 Modes of operation

The SWT supports three device modes of operation: normal, debug and stop. When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_CR *on page 815*. 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 in the SWT_CR *on page 815*. If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

40.2 External signal description

The SWT module does not have any external interface signals.

40.3 Memory map and register definition

The SWT programming model has seven 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. Other types of invalid accesses include: writes to read-only registers, incorrect values written to the service register when enabled, accesses to reserved addresses and accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid access, otherwise a bus error is generated. If either the HLK or

SLK bits in the SWT_CR are set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers are read-only.

The SWT memory map is shown in *Table 461*. The reset values of SWT_CR and SWT_TO are device specific. These values are determined by SWT inputs.

Table 461. SWT memory map

Offset	Register	Access	Reset Value	Location
0x0000	SWT Control Register (SWT_CR)	R/W	—(1)	*on page 815*
0x0004	SWT Interrupt Register (SWT_IR)	R/W	0x0000_0000	*on page 817*
0x0008	SWT Time-out Register (SWT_TO)	R/W	—(1)	*on page 817*
0x000C	SWT Window Register (SWT_WN)	R/W	0x0000_0000	*on page 818*
0x0010	SWT Service Register (SWT_SR)	W	0x0000_0000	*on page 818*
0x0014	SWT Counter Output Register (SWT_CO)	R	0x0000_0000	*on page 819*
0x0018	SWT Service Key Register (SWT_SK)	R/W	0x0000_0000	*on page 819*
0x001C-0x3FFF	Reserved			

1. The reset value for this register is device specific. See the device configuration chapter.

40.3.1 Register descriptions

The following sections detail the individual registers within the SWT programming model.

Note: See the memory map for the base address of SWT registers.

40.3.1.1 SWT Control Register (SWT_CR)

The SWT_CR contains fields for configuring and controlling the SWT. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0	0	0	0	0	0	0	0
W																
Reset(1)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	SMD		RIA	WND	ITR	HLK	SLK	0	STP	FRZ	WEN
W																
Reset(1)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

1. The reset value for the SWT_CR is device specific. See the device configuration section.

Figure 395. SWT Control Register (SWT_CR)

Table 462. SWT_CR field descriptions

Field	Name	Description
0-7	MAPn	Master Access Protection for Master n. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section 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

40.3.1.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Offset 0x0004 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 396. SWT Interrupt Register (SWT_IR)

Table 463. SWT_IR field descriptions

Field	Name	Description
31	TIF	Time-out Interrupt Flag. The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request. 1 Interrupt request due to an initial time-out.

40.3.1.3 SWT Time-Out Register (SWT_TO)

The SWT Time-Out (SWT_TO) register contains the 32-bit time-out period. The reset value for this register is device specific. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset 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	WTO							
W								
Reset(1)	* * * *	* * * *	* * * *	* * * *	* * * *	* * * *	* * * *	* * * *

Figure 397. SWT Time-Out Register (SWT_TO)

1. The reset value of the SWT_TO register is device specific. See the device configuration section.

Table 464. SWT_TO field descriptions

Field	Name	Description
0-31	WTO	Watchdog time-out period in clock cycles. An internal 32-bit down counter is loaded with this value when the service sequence is written or when the SWT is enabled. The initial value is 0x0003_FDE0. The minimum value is 0x0000_0100.

40.3.1.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset 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 398. SWT Window Register (SWT_WN)

Table 465. SWT_WN field descriptions

Field	Name	Description
0-31	WST	Window start value. When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

40.3.1.5 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) register is the target for service operation writes used to reset the watchdog timer.

Offset 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	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 399. SWT Service Register (SWT_SR)

Table 466. SWT_SR field descriptions

Field	Name	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[KEY] bit is set, two pseudorandom key values are written to service the watchdog, see Section *Figure 40.3.1.1: SWT Control Register (SWT_CR)* for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the soft lock bit (SWT_CR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field. When read, the WSC field always returns zero.

40.3.1.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

Offset 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	0	0	0	0	0	0	0	0	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 400. SWT Counter Output Register (SWT_CO)

Table 467. SWT_CO field descriptions

Field	Name	Description
0-31	CNT	Watchdog Count. When the watchdog is disabled (SWT_CR[WEN]=0), this field shows the value of the internal down counter. When the watchdog is enabled (SWT_CR[WEN]=1), this field is cleared (the value is 0x0000_0000). Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

40.3.1.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset 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 401. SWT Service Key Register (SWT_SK)

Table 468. SWT_SK field descriptions

Field	Name	Description
16-31	SK	Service Key.This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[KEY] is set, the next key value to be written to the SWT_SR is (17*SK+3) mod 2^{16}.

40.4 Functional description

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).

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[k]. 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.

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 reset value of the SWT_TO register is device specific.

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. The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset.The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register. 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.

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

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. 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 *Figure 402*. 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

k. See SWT section in device configuration chapter.

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.

Figure 402. Pseudorandom Key Generator

$$SK_{n+1} = (17*SK_n+3) \bmod 2^{16}$$

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.

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.

The interrupt then reset bit (SWT_CR[ITR]) 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.

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.

41 Periodic Interrupt Timer (PIT)

41.1 Introduction

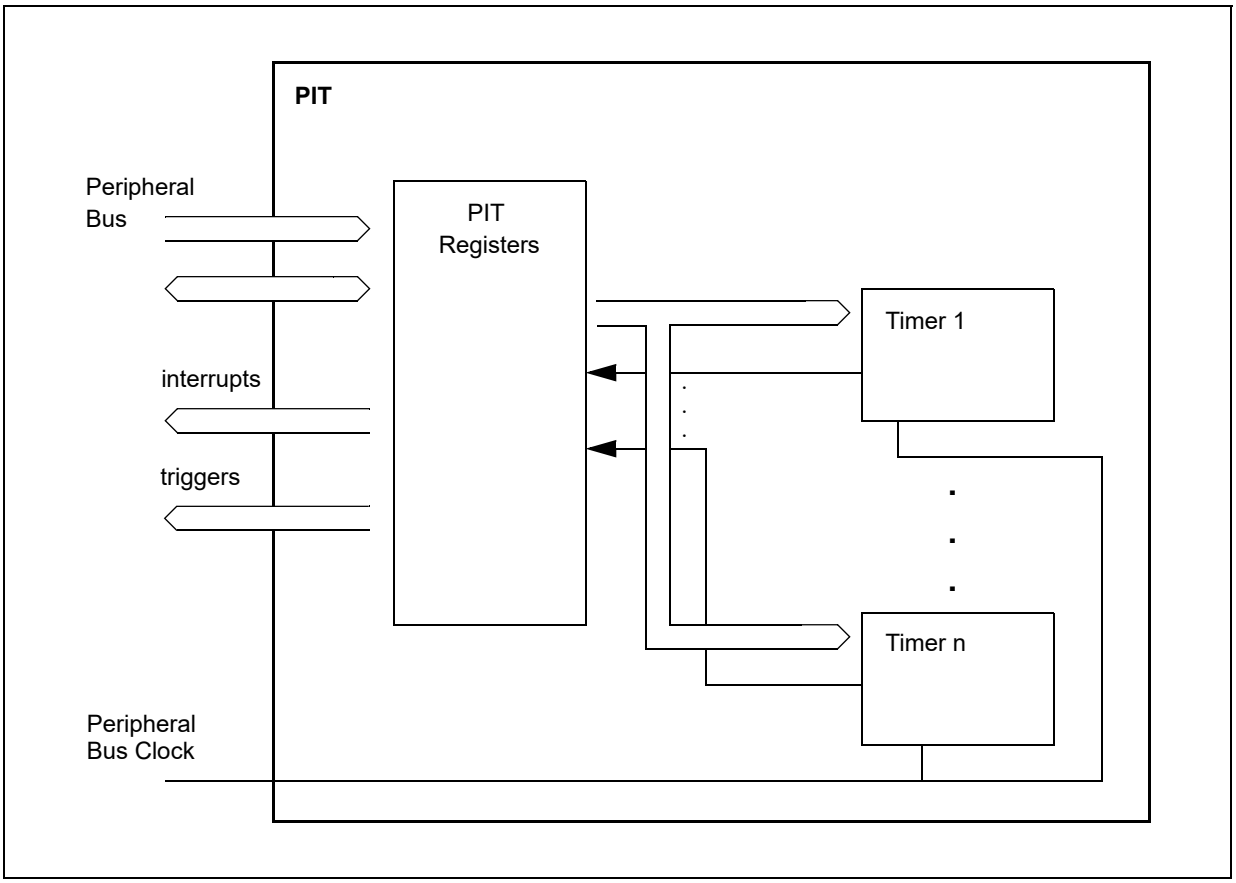
The PIT timer module is an array of timers that can be used to raise interrupts and trigger DMA channels.

Note: For the chip-specific implementation details of this module's instances see the Device Configuration chapter.

41.1.1 Block diagram

Table 403 shows the PIT block diagram.

Figure 403. Block diagram of the PIT



Note: Refer to the Chip Configuration information for the number of PIT channels used in this MCU.

41.1.2 Features

The main features of this block are:

- Timers can generate DMA trigger pulses
- Timers can generate interrupts
- All interrupts are maskable
- Independent timeout periods for each timer

41.2 Signal description

This module has no external pins.

41.2.1 Memory map/register description

This section provides a detailed description of all registers accessible in this module.

Note: Reserved registers will read as 0, writes will have no effect.

Refer to the Chip Configuration information for the number of PIT channels used in this MCU.

Table 469. PIT*x* memory map

Offset (hex)	Register name	Width (bits)	Access	Location
0000_0000	PIT Control Register (PIT*x*_MCR)	32	R/W	*on page 824*
0000_0004–0000_00DF	Reserved			
0000_00E0	PIT Upper Lifetime Timer Register (PIT*x*_LTMR64H)	32	R	*on page 825*
0000_00E4	PIT Lower Lifetime Timer Register (PIT*x*_LTMR64L)	32	R	*on page 825*
0000_00E8–0000_00FF	Reserved			
0000_0100	PIT Timer Load Value Register 0 (PIT*x*_LDVAL0)	32	R/W	*on page 826*
0000_0104	PIT Current Timer Value Register 0 (PIT*x*_CVAL0)	32	R	*on page 827*
0000_0108	PIT Timer Control Register 0 (PIT*x*_TCTRL0)	32	R/W	*on page 827*
0000_010C	PIT Timer Flag Register 0 (PIT*x*_TFLG0)	32	R/W	*on page 828*
0000_0110	PIT Timer Load Value Register 1 (PIT*x*_LDVAL1)	32	R/W	*on page 826*
0000_0114	PIT Current Timer Value Register 1 (PIT*x*_CVAL1)	32	R	*on page 827*
0000_0118	PIT Timer Control Register 1 (PIT*x*_TCTRL1)	32	R/W	*on page 827*
0000_011C	PIT Timer Flag Register 1 (PIT*x*_TFLG1)	32	R/W	*on page 828*
0000_0120	PIT Timer Load Value Register 2 (PIT*x*_LDVAL2)	32	R/W	*on page 826*
0000_0124	PIT Current Timer Value Register 2 (PIT*x*_CVAL2)	32	R	*on page 827*
0000_0128	PIT Timer Control Register 2 (PIT*x*_TCTRL2)	32	R/W	*on page 827*
0000_012C	PIT Timer Flag Register 2 (PIT*x*_TFLG2)	32	R/W	*on page 828*
0000_0130	PIT Timer Load Value Register 3 (PIT*x*_LDVAL3)	32	R/W	*on page 826*
0000_0134	PIT Current Timer Value Register 3 (PIT*x*_CVAL3)	32	R	*on page 827*

Table 469. PIT*x* memory map(Continued)

Offset (hex)	Register name	Width (bits)	Access	Location
0000_0138	PIT Timer Control Register 3 (PIT*x*_TCTRL3)	32	R/W	on page 827
0000_013C	PIT Timer Flag Register 3 (PIT*x*_TFLG3)	32	R/W	on page 828
0000_0140	PIT Timer Load Value Register 4 (PIT*x*_LDVAL4)	32	R/W	on page 826
0000_0144	PIT Current Timer Value Register 4 (PIT*x*_CVAL4)	32	R	on page 827
0000_0148	PIT Timer Control Register 4 (PIT*x*_TCTRL4)	32	R/W	on page 827
0000_014C	PIT Timer Flag Register 4 (PIT*x*_TFLG4)	32	R/W	on page 828
0000_0150	PIT Timer Load Value Register 5 (PIT*x*_LDVAL5)	32	R/W	on page 826
0000_0154	PIT Current Timer Value Register 5 (PIT*x*_CVAL5)	32	R	on page 827
0000_0158	PIT Timer Control Register 5 (PIT*x*_TCTRL5)	32	R/W	on page 827
0000_015C	PIT Timer Flag Register 5 (PIT*x*_TFLG5)	32	R/W	on page 828
0000_0160	PIT Timer Load Value Register 6 (PIT*x*_LDVAL6)	32	R/W	on page 826
0000_0164	PIT Current Timer Value Register 6 (PIT*x*_CVAL6)	32	R	on page 827
0000_0168	PIT Timer Control Register 6 (PIT*x*_TCTRL6)	32	R/W	on page 827
0000_016C	PIT Timer Flag Register 6 (PIT*x*_TFLG6)	32	R/W	on page 828
0000_0170	PIT Timer Load Value Register 7 (PIT*x*_LDVAL7)	32	R/W	on page 826
0000_0174	PIT Current Timer Value Register 7 (PIT*x*_CVAL7)	32	R	on page 827
0000_0178	PIT Timer Control Register 7 (PIT*x*_TCTRL7)	32	R/W	on page 827
0000_017C	PIT Timer Flag Register 7 (PIT*x*_TFLG7)	32	R/W	on page 828

41.2.1.1 PIT Module *x* Control Register (PIT*x*_MCR)

The PIT Module Control Register (PIT*x*_MCR) controls whether the timer clocks should be enabled and whether the timers should run in debug mode.

Address: Base + 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 404. PIT Module Control Register (PIT*x*_MCR)

Table 470. PIT*x*_MCR field descriptions

Field	Description
0–29 Reserved	This read-only bitfield is reserved and always has the value zero.
30 MDIS	Module Disable This is used to disable the module clock. This bit must be enabled before any other setup is done. 0 Clock for PIT timers is enabled. 1 Clock for PIT timers is disabled.
31 FRZ	Freeze Allows the timers to be stopped when the device enters debug mode. 0 Timers continue to run in debug mode. 1 Timers are stopped in debug mode.

41.2.1.2 PIT Module *x* Upper Lifetime Timer Register (PIT*x*_LTMR64H)

The PIT Module *x* Upper Lifetime Timer Register (PIT*x*_LTMR64H) is intended for applications that chain timer 0 and timer 1 to build a 64-bit life timer.

Address : Base + 0x00E0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LTH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LTH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 405. PIT Upper Lifetime Timer Register (PITx_LTMR64H)

Table 471. PITx_LTMR64H field descriptions

Field	Description
0–31 LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

41.2.1.3 PIT Module *x* Lower Lifetime Timer Register (PITx_LTMR64L)

The PIT Module *x* Lower Lifetime Timer Register (PIT*x*_LTMR64L) is intended for applications that chain timer 0 and timer 1 to build a 64-bit life timer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of

the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Address: Base + 0x00E4 Access: User read-only

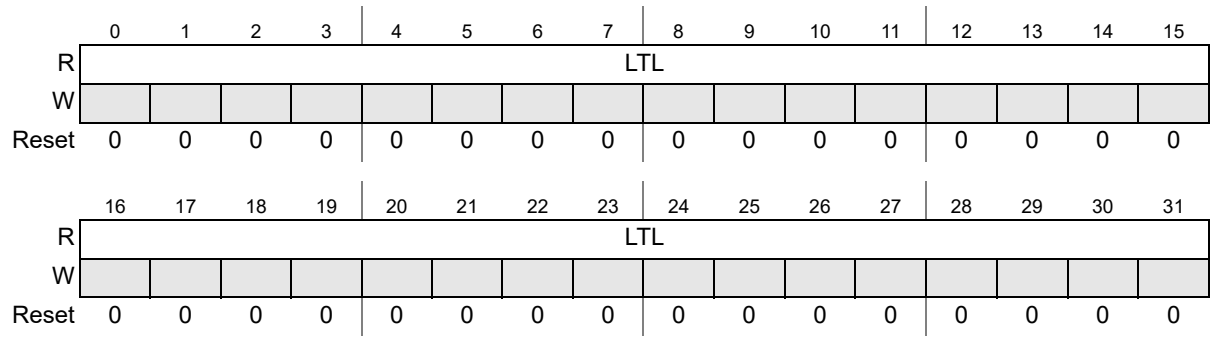


Figure 406. PIT Lower Lifetime Timer Register (PITx_LTMR64L)

Table 472. PITx_LTMR64L field descriptions

Field	Description
0–31 LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

41.2.1.4 PIT Module *x* Timer Load Value Register *n* (PIT*x*_LDVAL*n*)

Address: Base + 0x0100 Access: User read/write

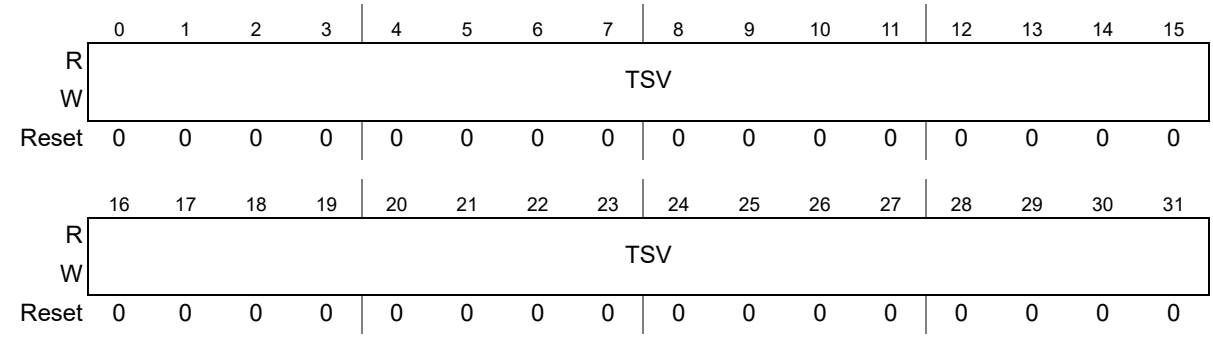


Figure 407. PIT Module *x* Timer Load Value Register *n* (PIT*x*_LDVAL*n*)

Table 473. PIT*x*_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.

41.2.1.5 PIT Module *x* Current Timer Value Register *n* (PIT*x*_CVAL*n*)

Address: Base + 0x0104 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 408. PIT Module *x* Current Timer Value Register *n* (PIT*x*_CVAL*n*)

Table 474. PIT*x*_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.

41.2.1.6 PIT Module *x* Timer Control Register *n* (PIT*x*_TCTRL*n*)

Address: Base + 0x0108 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 409. Timer Control Register (PIT*x*_TCTRL*n*)

Table 475. PIT*x*_TCTRL*n* field descriptions

Field	Description
0–28 Reserved	This read-only bitfield is reserved and always has the value zero.
29 CHN	Chain Mode Bit When activated, timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be changed. 0 Timer is not chained. 1 Timer is chained to previous timer (example, for channel 2 if this bit is set timer2 is chained 1).
30 TIE	Timer Interrupt Enable Bit 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set. When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 Timer n is disabled. 1 Timer n is active.

41.2.1.7 PIT Module *x* Timer Flag Register *n* (PIT*x*_TFLG*n*)

Address: Base + 0x010C 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 410. Timer Flag Register *n* (PIT*x*_TFLG*n*)

Table 476. PIT*x*_TFLG*n* field descriptions

Field	Description
0–30 Reserved	This read-only bitfield is reserved and always has the value zero.
31 TIF	Timer Interrupt Flag TIF is set to 1 at the end of the timer period. This flag can be cleared only by writing it with 1. Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

41.3 Functional description

This section provides the functional description of the module.

41.3.1 General

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses as well as to generate interrupts. Each interrupt is available on a separate interrupt line.

41.3.1.1 Timers

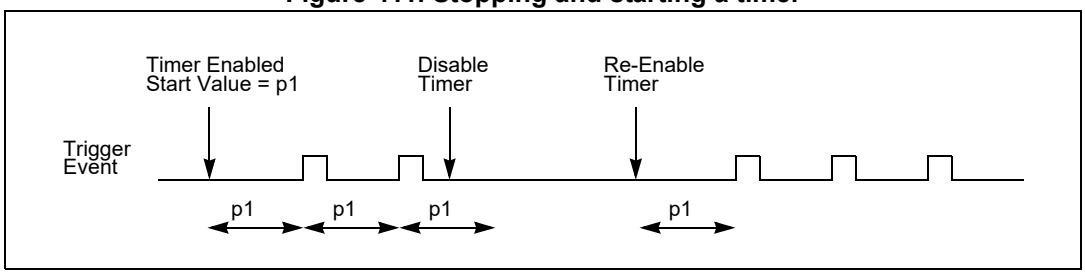
The timers generate triggers at periodic intervals, when enabled. They load their start values, as specified in their LDVAL registers, then count down until they reach 0. Then they load their respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked (by setting the TIE bits in the TCTRL registers). A new interrupt can be generated only after the previous one is cleared.

If desired, the current counter value of the timer can be read via the CVAL registers.

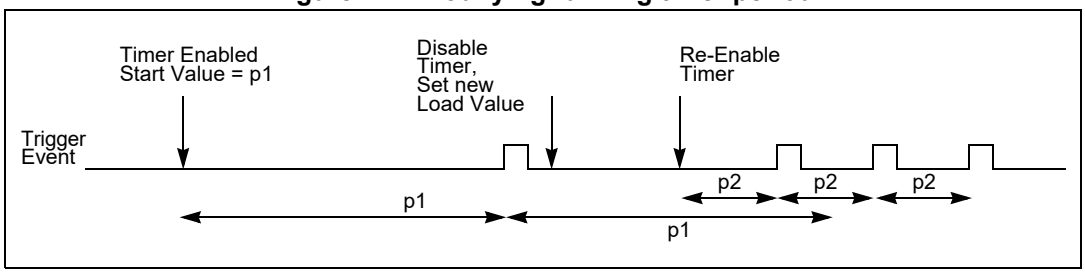
The counter period can be restarted, by first disabling, then enabling the timer with the TEN bit (see the following figure).

Figure 411. Stopping and starting a timer



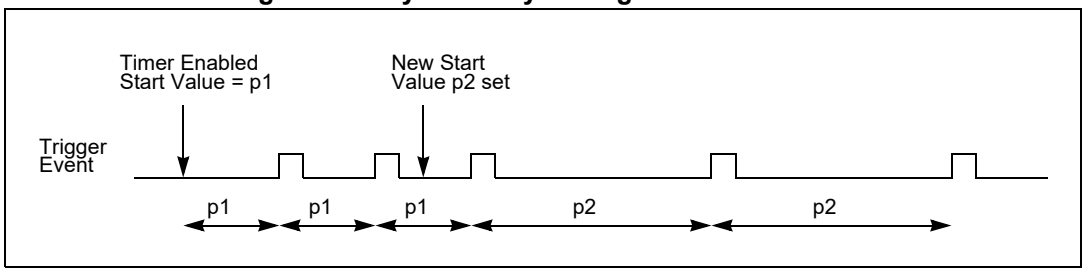
The counter period of a running timer can be modified, by first disabling the timer, setting a new load value and then enabling the timer again (see the following figure).

Figure 412. 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 413. Dynamically setting a new load value



41.3.1.2 Debug mode

In debug mode, the timers will be frozen based on field PIT*x*_MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system (for example, the timer values) and then continue the operation.

41.3.2 Interrupts

All of the timers support interrupt generation. Refer to the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting the TIE bits. The timer interrupt flags (TIF) are set to 1 when a timeout occurs on the associated timer, and are cleared to '0' by writing a '1' to that TIF bit.

41.3.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer *n*-1 has counted down to 0, counter *n* will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

41.4 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- Timer 1 creates an interrupt every 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 PITx_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 18.

```
// 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
```

41.5 Example configuration for chained timers

In the example configuration:

- The PIT clock has a frequency of 100 MHz.
- Timers 1 and 2 are available.
- An interrupt shall be raised every 1 hour.

First the PIT module needs to be activated by writing a ‘0’ to field PIT*x*_MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to timer 1 and programmed to trigger every 10 times.

The value for the LDVAL register trigger is calculated as number of cycles –1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for timer 2 is enabled by setting PIT*x*_TCTRL2[TIE], the chain mode is activated by setting CHN, and the timer is started by writing a ‘1’ to PIT*x*_TCTRL2[TEN]. In PIT*x*_TCTRL1, the TEN bit needs to be set, and CHN and TIE bits are cleared.

The following example code matches the described setup:

Example 19.

```
// turn on PIT
PIT_CTRL = 0x00;

// Timer 2
PIT_LDVAL2 = 0x00000009; // setup timer 2 for 10 counts
PIT_TCTRL2 = TIE; // enable timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain timer 2 to timer 1
PIT_TCTRL2 |= TEN; // start timer 2
```

```
// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup timer 1 for 600 000 000 cycles
PIPIT_TCTRL1 = TEN; // start timer 1
```

41.6 Example Configuration for the Lifetime Timer

To configure the life timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a ‘0’ to field PIT*x*_MCR[MDIS], then the LDVAL registers need to be set to the maximum value.

The following example code matches the described setup:

Example 20.

```
// turn on PIT
PIT_CTRL = 0x00;

// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0; // disable timer 2 interrupts
PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN; // start timer 1

// Timer 0
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0for maximum counting period
PIT_TCTRL0 = TEN; // start timer 0
```

To access the lifetime read first LTMR64H then LTMR64L.

Equation 21 **current_uptime = PIT_LTMR64H << 64;**
current_uptime = current uptime + PIT_LTMR64L

The timer is a down counter.

42 GTM Development Interface (GTMDI)

42.1 About this module

The GTMDI debug module unit is a hardware module that:

- Monitors the software and hardware activities of GTM-IP
- Reads the registers without interfering with the logic status, even for FIFOs, by configuring the debug access controls

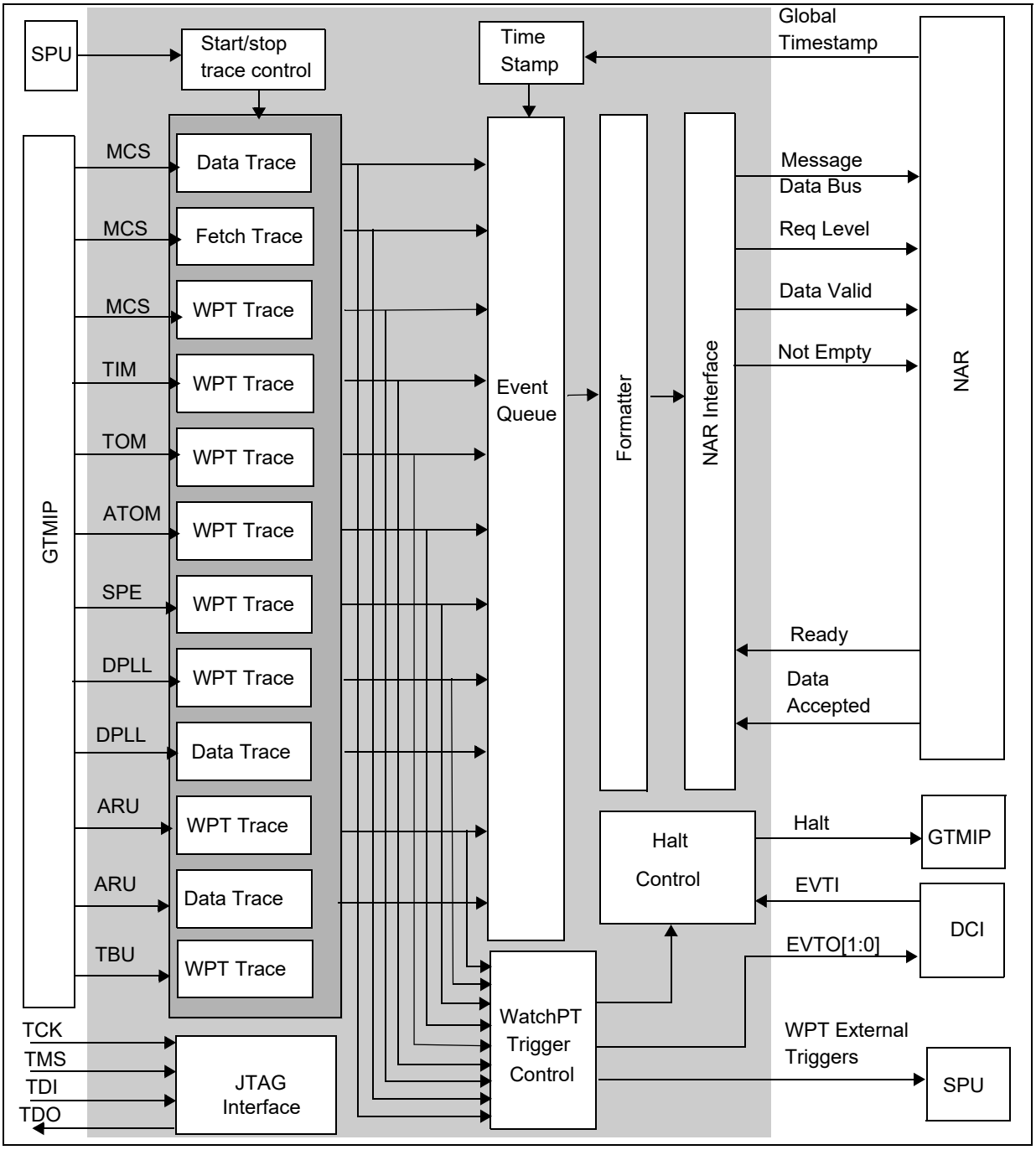
42.2 Introduction

The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including MCS cores, ARU data trace, Input Channel Filters trace, Output Timer channels, DPLL and SPE submodules. The GTMDI interfaces to the GTM IP using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules in order to implement complex debug features. Trace capability is also provided along with a global time-stamp that allows real time trace for several GTM submodule events.

The GTMDI implements a Nexus Client standard interface and a JTAG standard Port Controller to provide communication capability with the JTAG and NPC Aurora router on-chip modules. Note that the Nexus standard is only used for the physical interfaces. In order to implement the debug features such as breakpoint and messages required for the GTM debug, the Nexus standard could not be followed.

Figure 414 shows a block diagram of the Generic Timer Module (GTM) Development Interface.

Figure 414. GTMDI block diagram



42.3 Overview

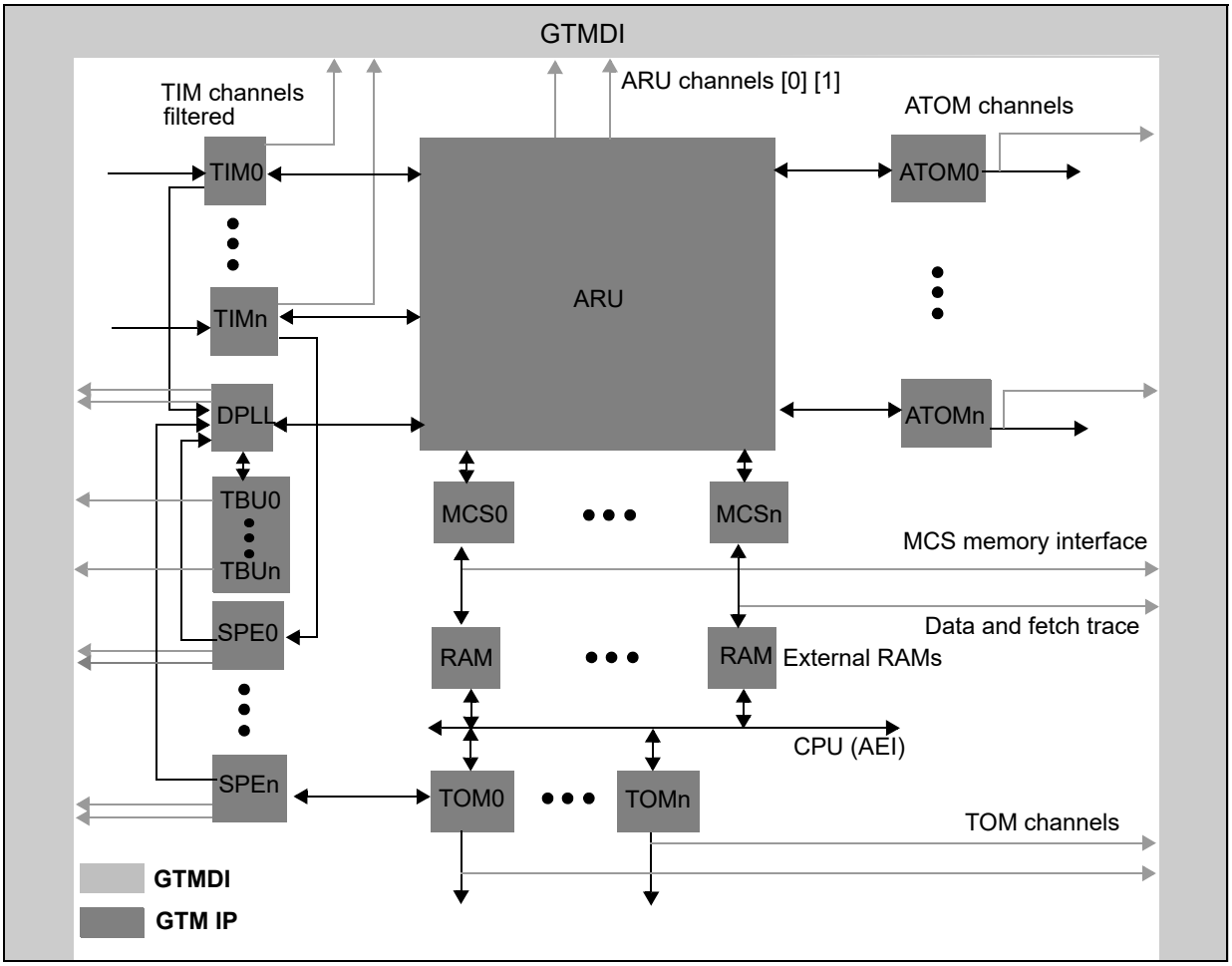
The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including Multi-Channel Sequencer (MCS) cores, Advanced Routing Unit (ARU) data trace, input channel filters trace, output timer channels, DPLL and Sensor Pattern Evaluation (SPE) sub-modules. The GTMDI interfaces to the GTM IP using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules to

implement complex debug features. Trace capability is also provided along with a global time-stamp which allows real time trace for several GTM sub-module events.

The GTMDI implements a Nexus Client standard interface and a JTAG standard Port Controller to provide communication capability with the JTAG and NAR on-chip modules. The Nexus standard is only used for the physical interfaces. Due to specific requirements for the GTM debug, the Nexus standard is not followed.

Figure 415 describes the interconnection between the GTMDI and GTM modules. The GTMDI selects which signal or bus to monitor since it does not have the capability to monitor all signals in parallel. The multiplexing architecture is described in the following sections.

Figure 415. GTMDI and GTM interface



Note: Figure 415 does not show all of the GTM internal modules. It shows only the modules that interface to the GTMDI debug module.

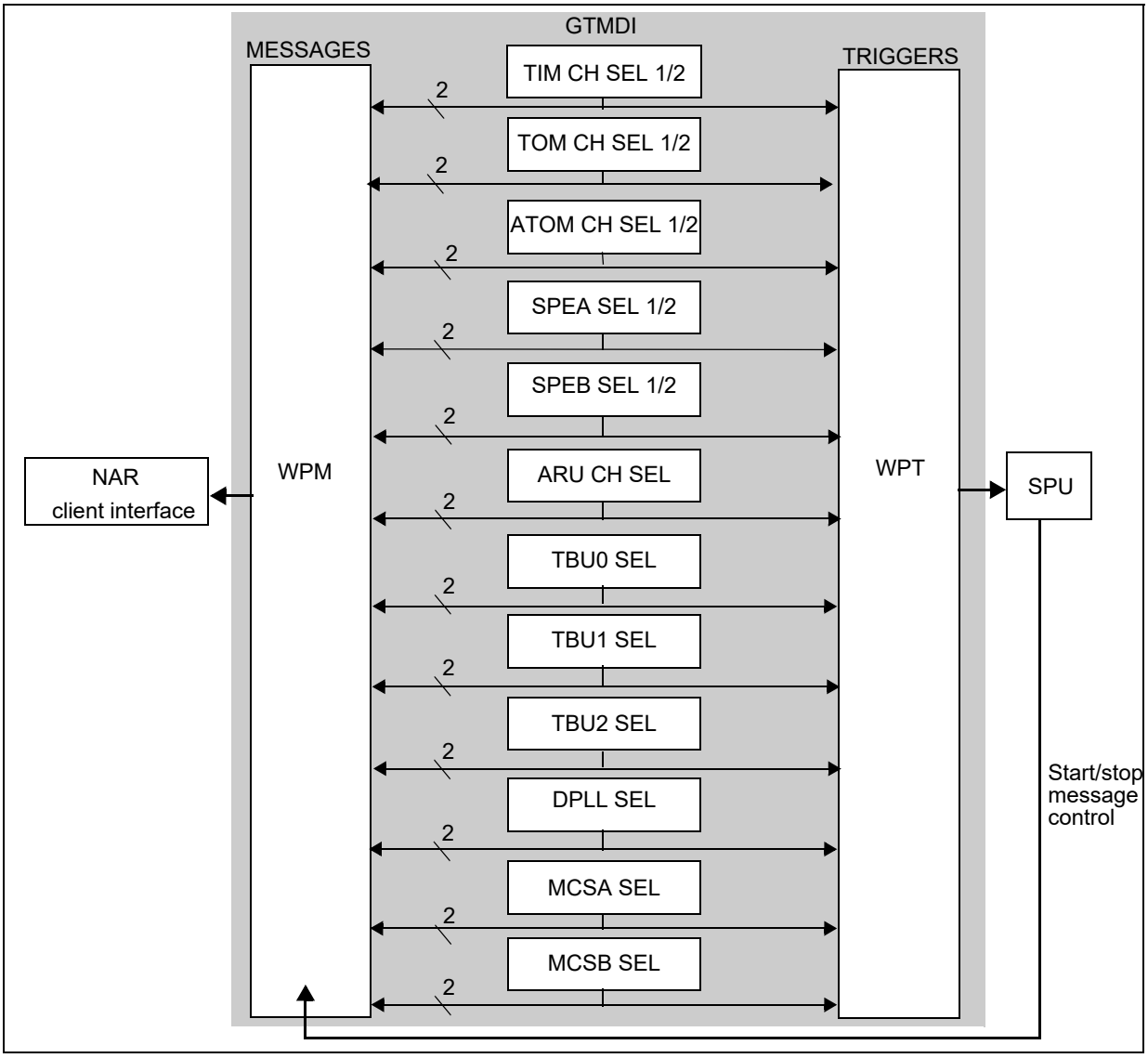
The number of ATOM, TIM, TOM, MCS and SPE modules in *Figure 415* is shown in *Table 477*.

Table 477. GTM Sub-module instances

Module	Number of instances
ATOM	3
TIM	2
TOM	2
MCS	numMCS
SPE	2

Figure 416 shows the Watchpoint Triggers (WPT) and Watchpoint Messages (WPM). Each GTM sub-module generates two watchpoints which can be routed internally to the GTMDI and generate watchpoint triggers or routed to the message formatter and generate watchpoint messages. The selection between watchpoint triggers and messages is implemented per sub-module. Messages and watchpoint triggers can also be generated at the same time based on the same event within a sub-module, thus a TIM channel transition can generate a watchpoint trigger and a watchpoint message. The watchpoint triggers may be sent to the Sequence Processing Unit (SPU) outside the GTMDI if enabled by appropriate control register bits for each sub-module, such as Timer Input Module (TIM), Timer Output Module (TOM) and ARU-connected Timer Output Module (ATOM). The watchpoint messages, if enabled, are sent to the NAR through the Nexus Client interface.

Figure 416. Watchpoint triggers and watchpoint messages



The SPU provides external control of the watchpoint messages generated by the GTMDI thus controlling the bandwidth over the Nexus Client interface. This functionality is implemented through Start/Stop control signals from the SPU. These signals are connected to the sub-module register bit that enables the message transmission implementing a dual control over these registers, either by JTAG interface programing or by SPU Start/Stop signals. See *Section 42.6, Functional description* for more details about register control and general architecture.

42.3.1 Features

The GTMDI block implements the following features:

- Full duplex pin interface for medium and high visibility throughput
  - 2 $\overline{\text{EVTO}}$ (Event Out) for GTM real time signal monitoring
  - 1 $\overline{\text{EVTI}}$ (Event In)
  - IEEE 1149.1 (JTAG) Test Access Port (TAP)
    - 4 pins (TDI, TDO, TMS, and TCK)
    - Reset input $\overline{\text{TRST}}$ driven by either the Nexus Port Controller or an external pin
- GTM Development Support
  - Read/Write registers in debug mode from JTAG port
  - Ability to enter debug mode at reset negation or during normal execution

Note: Enter debug out-of-reset is provided through the SoC system level debug control signal. In order to allow this functionality the GDE and SDBE control bits in the GTMDI_DC register should be set.

  - Halt and Watchpoint Configuration
  - MCS Data Trace Messaging (DTM) and Watchpoint Messaging (WPM) allows the development tool to trace reads and writes to selected MCS RAM address ranges
    - Two data trace windows for each MCS core with programmable address ranges and access attributes are provided. Data trace windowing reduces the requirement on the Aurora port bandwidth by constraining the number of trace locations.
    - Fetch trace capability controlled by two programmable Watchpoints.
    - Private messages are used to indicate special MCS core cases not covered by public messages. Two MCS cores can be debugged at the same time.

Note: Enabling two MCS cores at the same time for data trace can exceed the capacity of the NAR Client interface throughput thus information can be lost.

  - DPLL Data Trace Messaging and Watchpoint Trace Messaging allows the development tool to trace reads and writes to selected DPLL RAM addresses and selected data value.
  - ARU Data Trace Messaging and Watchpoint Trace Messaging allows the development tool to trace selected ARU channels.
  - Watchpoint Messaging (WPM)
- Trigger Control
  - The GTMDI provides selection of trigger signals that generate debug actions at system level such as Halt GTM itself or cores in the same device.
  - Trace enable signals are provided to control the start/stop of trace messages for each one of the GTM Nexus message sources, such as Timer Input Module (TIM), Timer Output Module (TOM), ARU-connected Timer Output Module (ATOM), etc.
- GTMDI clock operation
  - The module is able to operate in any frequency ratio considering GTMDI and JTAG frequency. The GTMDI operates from the same clock source as the GTM IP module.

42.3.2 Modes of operation

The GTMDI block is placed in reset when the Nexus reset signal is asserted. The JTAG registers controlled by the TCK clock are also put into the reset state if the TAP controller 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 initial TAP controller state. Asserting the Nexus reset signal results in asynchronous entry into the reset state not only of the TCK controlled registers but all registers in the GTMDI. The GTMDI 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 auxiliary output port enable outputs are negated
- The TDI, TMS, and TCK TAP inputs are ignored (The TMS and TCK signal are ignored only while the Nexus reset signal is asserted.)
- Registers default back to their reset values

Note: The GTMDI is not reset by the system reset signal. It is reset during device Power On reset or by a dedicated Nexus reset thus preserving internal registers state and programming during system reset.

Debug Mode can be controlled by on-chip logic through the system debug signal. The level on this signal defines if the GTM module is in debug mode, meaning that it is in Halt state and accepts write and read accesses to internal registers that do not modify the state of those registers. If asserted, the GTM enters Halt state. If negated, the GTM debug mode is controlled by other sources, internal to the GTMDI. The GTM may enter debug out of reset by asserting the system debug signal while the device is being reset. By detecting this condition, the GTMDI puts the GTM into the Halt state. The GDE and SDBE bits described in *Figure 418* must be set during reset allowing the GTM to enter debug state.

42.3.3 DATA port

GTMDI messages are sent through the Message Data Bus. The Message Data Bus has a fixed width of 32 bits which defines one BEAT of data. There are three BEATS per NAR clock thus 96 bits are transmitted in parallel.

42.4 External signal description

The GTMDI pinout provides interface for the transmission of messages and for accessing Nexus registers. The GTMDI pin definition is outlined in *Table 478*.

Table 478. GTMDI signal properties

Name	Port	Function	Reset State
EVTI	Auxiliary	Event In pin	—
$\overline{\text{EVTO}}$[1:0]	Auxiliary	Event Out pin	[11]
TCK	TAP	Test Clock Input	—
TDI	TAP	Test Data Input	—
TDO	TAP	Test Data Output	0
TMS	TAP	Test Mode Select Input	—
TRST	TAP	Test Reset Input	—

Each of the signals listed in *Table 478* is described in more detail in the following sections.

42.4.1 Event in ($\overline{\text{EVTI}}$)

Event In ($\overline{\text{EVTI}}$) is used to generate a Halt if enabled by the corresponding control bit. $\overline{\text{EVTI}}$ is an edge-sensitive signal. It is synchronized at the GTMDI input and a falling edge is detected.

42.4.2 Event out $\overline{\text{EVTO}}$[1:0]

Event Out ($\overline{\text{EVTO}}$) are two outputs from GTMDI for development tools with exact timing for a two watchpoints or Halt occurrence. $\overline{\text{EVTO}}$ is asserted for one system clock cycle when Halt or watchpoints are monitored. When internal GTM signals are monitored, such as TIM filtered signals or TOM/ATOM outputs, the $\overline{\text{EVTO}}$ represent the exact signal shape with a certain delay. In this monitoring mode the signal is not inverted through the $\overline{\text{EVTO}}$ output. The real time monitoring mode is intended for oscilloscope type of monitoring thus allowing the user to evaluate the actual shape of the signals. The source selection for the $\overline{\text{EVTO}}$ is centralized in the Development Control register GTMDI_DC.

42.4.3 Test clock input (TCK)

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the JTAG port.

42.4.4 Test data input (TDI)

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

42.4.5 Test data output (TDO)

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is actively driven in the SHIFT-IR and SHIFT-DR states of the JTAG state machine. TDO changes on the falling edge of TCK and is sampled on the rising edge of TCK.

42.4.6 Test Mode Select (TMS)

Test Mode Select (TMS) is an input pin used to sequence the JTAG state machine. TMS is sampled on the rising edge of TCK.

42.5 Register definition

This section provides a detailed description of all GTMDI registers accessible to the development tool. Individual bit-level descriptions and reset states of each register are included. *Table 479* shows the GTMDI registers and index values. These registers are not memory-mapped and can only be accessed via the JTAG interface.

Table 479. GTMDI registers

Register	Index	Read/write
Device Identity (DID)	0	R
Reserved	1–4	—
GTMDI Development Control (GTMDI_DC)	5	R/W
GTMDI Development Status (GTMDI_DS)	6	R
Reserved	7–10	—
TIM Watchpoint Control 1 (GTMDI_TIM_WPC1)	11	R/W
TIM Watchpoint Control 2 (GTMDI_TIM_WPC2)	12	R/W
Reserved	13–16	—
TOM Watchpoint Control 1 (GTMDI_TOM_WPC1)	17	R/W
TOM Watchpoint Control 2 (GTMDI_TOM_WPC2)	18	R/W
Reserved	19–22	—
ATOM Watchpoint Control 1(GTMDI_ATOM_WPC1)	23	R/W
ATOM Watchpoint Control 2 (GTMDI_ATOM_WPC2)	24	R/W
Reserved	25–28	—
SPEA Watchpoint Control 1(GTMDI_SPEA_WPC1)	29	R/W
SPEA Watchpoint Control 2(GTMDI_SPEA_WPC2)	30	R/W
Reserved	31–34	—
SPEB Watchpoint Control 1 (GTMDI_SPEB_WPC1)	35	R/W
SPEB Watchpoint Control 2 (GTMDI_SPEB_WPC2)	36	R/W
Reserved	37–40	—
DPLL Development Control (GTMDI_DPLL_WPC1)	41	R/W
DPLL Watchpoint Control 1 (GTMDI_DPLL_WPC2)	42	R/W
DPLL Watchpoint Control 2 (GTMDI_DPLL_WPC3)	43	R/W
DPLL Watchpoint Control 3 (GTMDI_DPLL_WPC4)	44	R/W
DPLL Watchpoint Control 4 (GTMDI_DPLL_WPC5)	45	R/W
DPLL Data Trace Control (GTMDI_DPLL_DTC)	46	R/W

Table 479. GTMDI registers(Continued)

Register	Index	Read/write
Reserved	47–49	—
ARU Watchpoint Control (GTMDI_ARU_WPC1)	50	R/W
ARU Watchpoint Control (GTMDI_ARU_WPC2)	51	R/W
ARU DATA0H (GTMDI_ARU_DATA0H)	52	R/W
ARU DATA0L (GTMDI_ARU_DATA0L)	53	R/W
ARU DATA1H (GTMDI_ARU_DATA1H)	54	R/W
ARU DATA1L (GTMDI_ARU_DATA1L)	55	R/W
ARU Data Trace Control (GTMDI_ARU_DTC)	56	R/W
Reserved	57–59	—
MCSA Development Control (GTMDI_MCSA_DC)	60	R/W
Reserved	61	—
MCSA Watchpoint Control (GTMDI_MCSA_WPC)	62	R/W
MCSA Program Fetch Trace Control (GTMDI_MCSA_PTC)	63	R/W
MCSA Data Trace Control (GTMDI_MCSA_DTC)	64	R/W
MCSA Watchpoint Address 1 (GTMDI_MCSA_WPA1)	65	R/W
MCSA Watchpoint Address 2 (GTMDI_MCSA_WPA2)	66	R/W
MCSA Watchpoint Data 1 (GTMDI_MCSA_WPD1)	67	R/W
MCSA Watchpoint Data 2 (GTMDI_MCSA_WPD2)	68	R/W
MCSA Program Trace Channel Enable (GTMDI_MCSA_CE)	69	R/W
Reserved	70	—
MCSA Data Trace Address Range 1 (GTMDI_MCSA_DTAR1)	71	R/W
MCSA Data Trace Address Range 2 (GTMDI_MCSA_DTAR2)	72	R/W
Reserved	73–76	—
MCSB Development Control (GTMDI_MCSB_DC)	77	R/W
Reserved	78	—
MCSB Watchpoint Control (GTMDI_MCSB_WPC)	79	R/W
MCSB Program Fetch Control (GTMDI_MCSB_PTC)	80	R/W
MCSB Data Trace Control (GTMDI_MCSB_DTC)	81	R/W
MCSB Watchpoint Address 1 (GTMDI_MCSB_WPA1)	82	R/W
MCSB Watchpoint Address 2 (GTMDI_MCSB_WPA2)	83	R/W
MCSB Watchpoint Data 1 (GTMDI_MCSB_WPD1)	84	R/W
MCSB Watchpoint Data 2 (GTMDI_MCSB_WPD2)	85	R/W
MCSB Program Trace Channel Enable (GTMDI_MCSB_CE)	86	R/W
Reserved	87	—
MCSB Data Trace Address Range 1 (GTMDI_MCSB_DTAR1)	88	R/W

Table 479. GTMDI registers(Continued)

Register	Index	Read/write
MCSB Data Trace Address Range 2 (GTMDI_MCSB_DTAR2)	89	R/W
Reserved	90–93	—
TBU0 Watchpoint Control 1 (GTMDI_TBU0_WPC1)	94	R/W
TBU0 Watchpoint Control 2 (GTMDI_TBU0_WPC2)	95	R/W
TBU0 DATA (GTMDI_TBU0_DATA)	96	R/W
Reserved	97–100	—
TBU1 Watchpoint Control 1 (GTMDI_TBU1_WPC1)	101	R/W
TBU1 Watchpoint Control 2 (GTMDI_TBU1_WPC2)	102	R/W
TBU1 DATA (GTMDI_TBU1_DATA)	103	R/W
Reserved	104–107	—
TBU2 Watchpoint Control 1 (GTMDI_TBU2_WPC1)	108	R/W
TBU2 Watchpoint Control 2 (GTMDI_TBU2_WPC2)	109	R/W
TBU2 DATA (GTMDI_TBU2_DATA)	110	R/W

42.5.1 Register descriptions

This section consists of GTMDI register descriptions. The registers are shown in *Table 479*. The MCS cores selected for debugging are named MCSA and MCSB thus two MCS cores can be debugged at the same time. See registers GTMDI_MCSA_DC and GTMDI_MCSB_DC which specify the real number of the MCS associated with MCSA and MCSB.

42.5.1.1 Device identity register (DID)

The DID register contains the unique identification information as shown in *Figure 417*.

Register index: 0

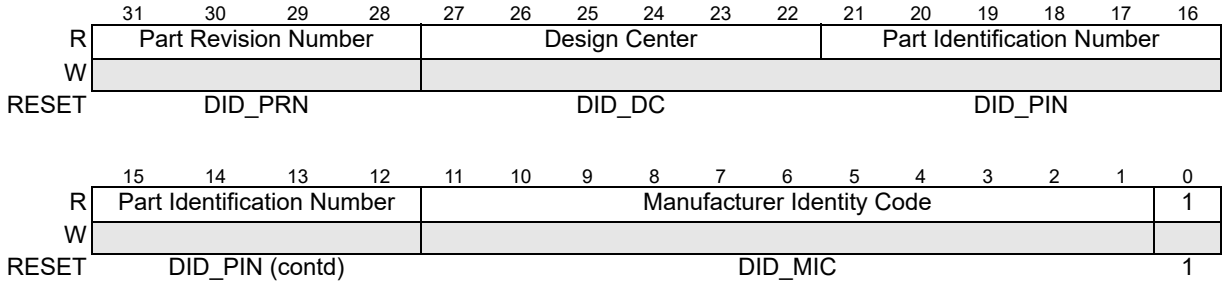


Figure 417. Device identity register (DID)

Table 480 describes the device identification register functions.

Table 480. DID field descriptions

Field	Description
31–28 PRN	Part Revision Number. Contains the revision number of the part.
27–22 DC	Design Center. Indicates the design center.
21–12 PIN	Part Identification Number. Contains the part number of the device. Value is PIN.
11–1 MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x020.
0 IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

42.5.1.2 GTMDI development control register (GTMDI_DC)

The GTMDI_DC register controls various trace and debug features of the GTM modules as described in *Figure 418*.

Register index: 5

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GDE	0	0	0	EOS0				0	0	0	0	EOS1			
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	EIDR	0	DBE	DBR	SDBE	0	TSFT
W								CHR								
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 418. GTMDI development control register (GTMDI_DC)

Table 481 describes the GTMDI_DC register functions.

Table 481. GTMDI_DC field descriptions

Field	Description
31 GDE	Global Debug Enable. This control bit disables all debug features of the GTMDI module including ability to put GTM IP in Halt state, modify EVTO output, send messages to the DATA bus and set triggers to the SPU module. Before changing the GTMDI configuration it is recommended to clear GDE bit to prevent spurious behavior of the module outputs. See *Figure 463* for details about debug enable logic. 0 GTMDI Global debug is disabled 1 GTMDI Global debug is enabled
27–24 EOS0	$\overline{\text{EVTO0}}$ Output Selection. This signal is dedicated to sel 1 which is the selection 1 from the indicated GTM modules. See the control registers for each module for a description of this selection. For TIM, TOM and ATOM selections the input filtered signal or output channel signal are used instead of the triggers. 0000 No event on output $\overline{\text{EVTO0}}$ remains at one 0001 Halt Indication, If $\overline{\text{EVTO0}}$ is zero indicates that GTM is halted 0010 TIM, Map EVTO0 from TIM channel filter output using ssel 1 selection control 0011 TOM, Map EVTO0 from TOM channel output using ssel1 selection control 0100 ATOM, Map EVTO0 from ATOM channel output using ssel 1 selection control 0101 SPEA, Map EVTO0 from SPEA, controlled by TSS1 described in *Figure 426* 0110 SPEB, Map EVTO0 from SPEB, controlled by TSS1 described in *Figure 428* 0111 ARU CH0, Map EVTO0 from ARU CH0I 1000 DPLL, Map EVTO0 from DPLL according to TSEL1 in *Figure 430* 1001 MCSA, Map EVTO0 from MCSA MCSA_sel selection control 1010 MCSB, Map EVTO0 from MCSB MCSB_sel selection control 1011 TBU0 Watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU0_WPC1 register 1100 TBU1 watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU1_WPC1 register 1101 TBU2 watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU2_WPC1 register 1110 Reserved 1111 Reserved
19–16 EOS1	$\overline{\text{EVTO1}}$ Output Selection. This signal is dedicated to sel 2 which is the selection 2 from the indicated GTM modules. See the control registers for each module for a description of this selection. For TIM, TOM and ATOM selections the input filtered signal or output channel signal are used instead of the triggers. 0000 No event on output $\overline{\text{EVTO1}}$ remains at one 0001 Halt Indication, If $\overline{\text{EVTO1}}$ is zero indicates that GTM is halted 0010 TIM, Map EVTO1 from TIM channel filter output using ssel 1 selection control 0011 TOM, Map EVTO1 from TOM channel output using ssel1 selection control 0100 ATOM, Map EVTO1 from ATOM channel output using ssel 1 selection control 0101 SPEA, Map EVTO1 from SPEA, controlled by TSS2 described in *Figure 426* 0110 SPEB, Map EVTO1 from SPEB, controlled by TSS2 described in *Figure 428* 0111 ARU CH1, Map EVTO1 from ARU CH1 1000 DPLL, Map EVTO1 from DPLL according to TSEL2A/B in *Figure 430* 1001 MCSA, Map EVTO1 from MCSA MCSA_sel selection control 1010 MCSB, Map EVTO1 from MCSB MCSB_sel selection control 1011 TBU0 Watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU0_WPC1 register 1100 TBU1 watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU1_WPC1 register 1101 TBU2 watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU2_WPC1 register 1110 Reserved 1111 Reserved

Table 481. GTMDI_DC field descriptions(Continued)

Field	Description
8 CHR	Clear Halt Request. Writing this bit to one clears halt requests., which means GTM resumes normal operation out of halt state. The status bits of the GTMDI_DS register are cleared. This bit is self-cleared thus read always as zero. 0 No action 1 Clear halt requests
6 EIDR	EVTI Debug Request enable. The EIDR bit enables EVTI input to request the GTM to enter Halt state. After the GTM enters Halt state due to a pulse in EVTI signal, the EIDR bit state is not meaningful, thus it may be cleared and the GTM continues in Halt state. See *Figure 463* for details about debug enable logic. 0 Disable EVTI to control GTM to enter Halt state 1 Enable EVTI to control GTM to enter Halt state
4 DBE	Debug Enable. DBE enables debug mode thus allowing the GTM to enter debug mode which means halt regular operation. The debug events controlled by this register are the ones generated internally to the GTMDI, excluding ipg_debug and EVTI which have their dedicated enable control. See *Figure 463* for details about debug enable logic. 0 Debug mode disabled 1 Debug mode enabled
3 DBR	Debug Request. Mechanism to allow development tool to request modules to enter debug mode directly. See *Figure 463* for details about debug enable logic. 0 Does not modify GTM IP debug state 1 Request GTM IP to enter debug mode
2 SDBE	System Debug Enable. The SDBE bit enables the GTM to enter halt state if system level debug enable signal is asserted. This bit allows the debug controller to select specific sub-groups of cores or modules to halt, while others continue to operate. GDE bit must also be set to allow GTM to halt. See *Figure 463* for details about debug enable logic. 0 GTM does not enter halt state due to system debug enable signal assertion 1 GTM is enabled to enter halt state due to system debug enable signal assertion
0 TSFT	Timestamp Force Transmission. Forces the transmission of the Timestamp value along with the Nexus message data for all messages transmitted by the GTMDI Client module. When Timestamp inclusion is not forced into the client message, its inclusion depends upon the NAR settings. 0 Does not force transmission of Timestamp value 1 Forces transmission of Timestamp along with Nexus message

42.5.1.3 GTMDI development status register (GTMDI_DS)

The GTMDI_DS register described in *Figure 419* shows the status of various conditions that impact GTM development support. All status bits are dynamic and do not require clearing. Debug Status Event is queued due to a change in the Halt or Stop modes and a Debug Status Message is generated as soon as there is an opportunity in the NAR interface. If any status bit changes before the Status message is sent, this change is reflected in the message. The GTMDI_DS register dynamically tracks the debug condition status even when several of them are simultaneously set.

Register index: 6

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HLT	0	0	HS1												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STP	0	0	HS2												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 419. GTMDI development status register (GTMDI_DS)

Table 482 describes the GTMDI_DS register functions.

Note: The HS1 and HS2 bit fields are cleared as soon as the GTM resumes normal operation which is controlled by the CHR bit in the GTMDI_DC register. One exception is HS1[0] that indicates system debug request, which is not cleared by CHR. This bit is cleared when system debug request is negated.

Table 482. GTMDI_DS field descriptions

Field	Description
31 HLT	Halt Indication. The HLT bit indicates if the GTM IP is in Halt state, ready to receive debug accesses. 0 GTM is running 1 GTM is in Halt state
28–16 HS1	Halt Status1. The HS1 field shows which source generated the halt condition. More than one source can generate the halt condition at the same time. This register is only meaningful if the GTM is halted. Its content indicates the halt sources that occurred immediately when the halt mode is entered. 0000000000000No Trigger generated 1xxxxxxxxxxxxTIM x1xxxxxxxxxxxTOM xx1xxxxxxxxxxATOM xxx1xxxxxxxxxSPEA xxxx1xxxxxxxxSPEB xxxxx1xxxxxxxARU xxxxxx1xxxxxxDPLL xxxxxxx1xxxxxMCSA xxxxxxxx1xxxxMCSB xxxxxxxxx1xxxTBU0 Watchpoint 1 xxxxxxxxxx1xxTBU1 Watchpoint 1 xxxxxxxxxxx1xTBU2 Watchpoint 1 xxxxxxxxxxxx1External through system debug enable signal

Table 482. GTMDI_DS field descriptions(Continued)

Field	Description
15 STP	Stop Status. Indicates if the GTM is in stop mode with no active clocks. Besides the system level stop mode mechanism, the GTM may independently be programmed for STOP mode by setting the Module Disable Control bit. Thus, the STP field indicates if the GTM is in stop mode no matter if due to the system stop mode assertion or due to the MDIS bit in the GTM module control register. 0 GTM not stopped 1 GTM is stopped in low power mode
12–0 HS2	Halt Status2. Similarly to HS1 this field indicates the source for the halt condition selection 2. See *Figure 464: TIM selection logic* for reference about source selection 1 and 2. The HS2 field shows which source generated the halt condition 2. More than one source can generate the halt condition at the same time. This register is only meaningful if the GTM is halted. Its content indicates the halt condition source that occurred when the halt mode is entered. 0000000000000No Trigger generated 1xxxxxxxxxxxTIM x1xxxxxxxxxxTOM xx1xxxxxxxxxATOM xxx1xxxxxxxxSPEA xxxx1xxxxxxxSPEB xxxxx1xxxxxxARU xxxxxx1xxxxxDPLL xxxxxxx1xxxxMCSA xxxxxxxx1xxxMCSB xxxxxxxxx1xxxTBU0 Watchpoint 2 xxxxxxxxxx1xxTBU1 Watchpoint 2 xxxxxxxxxxx1xTBU2 Watchpoint 2 xxxxxxxxxxxx1$\overline{\text{EVTI}}$

42.5.1.4 TIM watchpoint control register 1 (GTMDI_TIM_WPC1)

The GTMDI_TIM_WPC1 register shown in *Figure 420* controls the Watchpoint Triggers WPT issued by TIM channels. These triggers are used by the GTM internal modules in order to control GTM Halt mode entering and also at SoC level. See *Figure 464* which describes the use of this register.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG. The SEN1 and SEN2 control bits from the GTMDI_TIM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 11

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	SSEL1			HEN1	WMC1	0	0	0	CHSEL1		
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	SSEL2			HEN2	WMC2	0	0	0	CHSEL2		
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 420. TIM watchpoint control 1 register (GTMDI_TIM_WPC1)

Table 483 describes the GTMDI_TIM_WPC1 register functions.

Table 483. GTMDI_TIM_WPC1 field descriptions

Field	Description
29–28 TSS1	TIM Channel Slope Selection 1. The TSS1 field selects the slope for the TIM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
26–24 SSEL1	TIM sub-module Source Selection 1. Selects which TIM sub-module is the source for watchpoint generation. 000 TIM0 001 TIM1 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected TIM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TIM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TIM Filter output to generate a Watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

Table 483. GTMDI_TIM_WPC1 field descriptions(Continued)

Field	Description
18–16 CHSEL1	TIM channel selection 1. Selects which channel within a selected TIM sub-module generates watchpoints. The selection is actually on the TIM channel input filter. 000 TIM CH0 001 TIM CH1 010 TIM CH2 011 TIM CH3 100 TIM CH4 101 TIM CH5 110 TIM CH6 111 TIM CH7
13–12 TSS2	TIM Channel Slope Selection 2. The TSS2 field selects the slope for the TIM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
10–8 SSEL2	TIM sub-module Source Selection 2. Selects which TIM sub-module is the source for watchpoint generation. 000 TIM0 001 TIM1 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
7 HEN2	Halt Enable 2. The HEN1 Halt Enable 2 bit controls if the selected TIM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TIM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC1 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TIM Filter output to generate a Watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
2–0 CHSEL2	TIM channel selection 2. Selects which channel within a selected TIM sub-module generates watchpoints. The selection is actually on the TIM channel input filter. 000 TIM CH0 001 TIM CH1 010 TIM CH2 011 TIM CH3 100 TIM CH4 101 TIM CH5 110 TIM CH6 111 TIM CH7

42.5.1.5 TIM watchpoint control register 2 (GTMDI_TIM_WPC2)

The GTMDI_TIM_WPC2 register shown in *Figure 421* controls the Watchpoint Triggers WPT issued by TIM channels. These watchpoint triggers are used to control GTM Halt state entering, watchpoint messages and control external logic to the GTMDI. See *Figure 464* for details of the logic that uses the bits in this register.

Register index: 12

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 421. TIM watchpoint control 2 register (GTMDI_TIM_WPC2)

Table 484 describes the GTMDI_TIM_WPC2 register functions.

Note: The SHARED_WPT and SHARED_WPT_ST signals mentioned in Table 484 and in other tables in this document, refer to the GTMDI top level connections. They correspond to Watchpoint trigger outputs of GTMDI and Stop/Start input signals respectively.

Table 484. GTMDI_TIM_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TIM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued. This enable refers to the output selected by the WTSEL1 field. 0 Disable Watchpoint Trigger Generation 1 Enable Watchpoint Trigger Generation

Table 484. GTMDI_TIM_WPC2 field descriptions(Continued)

Field	Description
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected TIM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TIM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TIM Channel. Other modules can select the same watchpoint trigger, in this case the result signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used to control the watchpoint trace messages for the selected TIM channel. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected TIM channel. This enable refers to the output selected by the WTSEL2 field. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TIM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.6 TOM watchpoint control register 1 (GTMDI_TOM_WPC1)

The GTMDI_TOM_WPC1 register shown in *Figure 422* controls the Watchpoint Triggers issued by TOM channels. These Watchpoint triggers are used to control the GTM requests to enter Halt, issue watchpoint messages and also by logic external to the GTMDI at SoC level.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TOM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 17

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	SSEL1			HEN1	WMC1	0	0	CHSEL1			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	SSEL2			HEN2	WMC2	0	0	CHSEL2			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 422. TOM watchpoint control 1 register (GTMDI_TOM_WPC1)

Table 485 describes the GTMDI_TOM_WPC1 register functions.

Table 485. GTMDI_TOM_WPC1 field descriptions

Field	Description
29–28 TSS1	TOM Channel Slope Selection 1. The TSS1 field selects the slope for the TOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
26–24 SSEL1	TOM sub-module Source Selection 1. Selects which TOM sub-module is the source for watchpoint generation. 000 TOM0 001 TOM1 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected TOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

Table 485. GTMDI_TOM_WPC1 field descriptions(Continued)

Field	Description
19–16 CHSEL1	TOM channel selection 1. Selects which channel within a selected TOM sub-module generates watchpoints. 0000 TOM CH0 0001 TOM CH1 0010 TOM CH2 0011 TOM CH3 0100 TOM CH4 0101 TOM CH5 0110 TOM CH6 0111 TOM CH7 1000 TOM CH8 1001 TOM CH9 1010 TOM CH10 1011 TOM CH11 1100 TOM CH12 1101 TOM CH13 1110 TOM CH14 1111 TOM CH15
13–12 TSS2	TOM Channel Slope Selection 2. The TSS2 field selects the slope for the TOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
10–8 SSEL2	TOM sub-module Source Selection 2. Selects which TOM sub-module is the source for watchpoint generation. 000 TOM0 001 TOM1 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
7 HEN2	Halt Enable 2. The HEN1 Halt Enable 2 bit controls if the selected TOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM

Table 485. GTMDI_TOM_WPC1 field descriptions(Continued)

Field	Description
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start and Stop signals inputs to the GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
3–0 CHSEL2	TOM channel selection 2. The CHSEL2[2:0] field selects a channel within the TOM selected sub-module to generate watchpoints. 0000 TOM CH0 0001 TOM CH1 0010 TOM CH2 0011 TOM CH3 0100 TOM CH4 0101 TOM CH5 0110 TOM CH6 0111 TOM CH7 1000 TOM CH8 1001 TOM CH9 1010 TOM CH10 1011 TOM CH11 1100 TOM CH12 1101 TOM CH13 1110 TOM CH14 1111 TOM CH15

42.5.1.7 TOM watchpoint control 2 register (GTMDI_TOM_WPC2)

The GTMDI_TOM_WPC2 register shown in *Figure 423* controls the Watchpoint Triggers issued by TOM channels. These watchpoints are used to force GTM into Halt state, generate watchpoint messages and control logic external to the GTMDI.

Register index: 18

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 423. TOM watchpoint control 2 register (GTMDI_TOM_WPC2)

Table 486 describes the GTMDI_TOM_WPC2 register functions.

Table 486. GTMDI_TOM_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TOM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued by the TOM Channel output selected by WTSEL1. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected TOM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available Watchpoint trigger outputs which one is connected to the TOM selected channel. Other modules can select the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 486. GTMDI_TOM_WPC2 field descriptions(Continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected TOM channel. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs of the GTMDI to control Watchpoint trace messages generated by the TOM selected channel. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.8 ATOM watchpoint control 1 register (GTMDI_ATOM_WPC1)

The GTMDI_ATOM_WPC1 register shown in *Figure 424* controls the Watchpoint Triggers issued by ATOM channels. These watchpoint triggers are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_ATOM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 23

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		SSEL1				HEN1	WMC1	0	0	0	CHSEL1		
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		SSEL2				HEN2	WMC2	0	0	0	CHSEL2		
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 424. ATOM watchpoint control 1 register (GTMDI_ATOM_WPC1)

Table 487 describes the GTMDI_ATOM_WPC1 register functions.

Table 487. GTMDI_ATOM_WPC1 field descriptions

Field	Description
29–28 TSS1	ATOM Channel Slope Selection 1. The TSS1 field selects the slope for the ATOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
27–24 SSEL1	ATOM sub-module Source Selection 1. Selects which ATOM sub-module is the source for watchpoint generation. 0000 ATOM0 0001 ATOM1 0010 ATOM2 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
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected ATOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected ATOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected ATOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
18–16 CHSEL1	ATOM channel selection 1. Selects which channel in an ATOM module generates watchpoints. 000 ATOM CH0 001 ATOM CH1 010 ATOM CH2 011 ATOM CH3 100 ATOM CH4 101 ATOM CH5 110 ATOM CH6 111 ATOM CH7
13–12 TSS2	ATOM Channel Slope Selection 2. The TSS2 field selects the slope for the ATOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved

Table 487. GTMDI_ATOM_WPC1 field descriptions(Continued)

Field	Description
11–8 SSEL2	ATOM sub-module Source Selection 2. Selects which ATOM module is the source for watchpoint generation. 0000 ATOM0 0001 ATOM1 0010 ATOM2 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
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the ATOM selected channel is enabled to HALT the GTM module. If this bit is set and an event occurs on the ATOM selected channel, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected ATOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
2–0 CHSEL2	ATOM channel selection 2. The CHSEL2[2:0] field selects a channel within the TOM selected sub-module to generate watchpoints. 000 ATOM CH0 001 ATOM CH1 010 ATOM CH2 011 ATOM CH3 100 ATOM CH4 101 ATOM CH5 110 ATOM CH6 111 ATOM CH7

42.5.1.9 ATOM watchpoint control 2 register (GTMDI_ATOM_WPC2)

The GTMDI_ATOM_WPC2 register shown in *Figure 424* controls the Watchpoint Triggers issued by ATOM channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 24

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 425. ATOM watchpoint control 2 register (GTMDI_ATOM_WPC2)

Table 488 describes the GTMDI_ATOM_WPC2 register functions.

Table 488. GTMDI_ATOM_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected ATOM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued by the ATOM Channel output selected by WTSEL1. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected ATOM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_ATOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 488. GTMDI_ATOM_WPC2 field descriptions(Continued)

Field	Description
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available Watchpoint trigger outputs which one is connected to the ATOM selected channel. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected ATOM channel. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs of the GTMDI to control Watchpoint trace messages generated by the ATOM selected channel. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_ATOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.10 SPEA watchpoint control 1 register 1 (GTMDI_SPEA_WPC1)

The GTMDI_SPEA_WPC1 register shown in *Figure 426* controls the Watchpoint Triggers issued by SPEA channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and to control logic outside of the GTMDI module, such as SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_SEP0_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 29

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	SSEL			0	WMC1	TSS1		0	0	HEN1	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	WMC2	TSS2		0	0	HEN2	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 426. SPEA watchpoint control 1 register (GTMDI_SPEA_WPC1)

Table 489 describes the GTMDI_SPEA_WPC1 register functions.

Table 489. GTMDI_SPEA_WPC1 field descriptions

Field	Description
26–24 SSEL	SPEA sub-module Source Selection. Selects which SPE sub-module is the source for watchpoint generation. 000 SPE0 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEA NIPD to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
21–20 TSS1	SPEA NIPD Slope Selection. The TSS1 field selects the slope for the NIPD signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	Halt Enable 1. The HEN1 Halt Enable 1bit controls if the SPEA NIPD active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEA DIR output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

Table 489. GTMDI_SPEA_WPC1 field descriptions(Continued)

Field	Description
5–4 TSS2	SPEA DIR Slope Selection. The TSS2 field selects the slope for the DIR signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
1 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the SPEA DIR active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM

42.5.1.11 SPEA watchpoint control 2 register (GTMDI_SPEA_WPC2)

The GTMDI_SPEA_WPC2 register shown in *Figure 427* controls the Watchpoint Triggers issued by SPEA. These watchpoints are used for GTM to enter Halt state, generate watchpoint trace messages and to control logic outside the GTMDI module, such as SPU interface.

Register index: 30

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 427. SPEA watchpoint control 2 register (GTMDI_SPEA_WPC2)

Table 490 describes the GTMDI_SPEA_WPC2 register functions.

Table 490. GTMDI_SPEA_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available watchpoint triggers which one is connected to the NIPD SPEA signal. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the SPEA NIPD selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the SPEA NIPD selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_SPEA_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DIR SPEA signal. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 490. GTMDI_SPEA_WPC2 field descriptions(Continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the SPEA DIR selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the SPEA DIR selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_SPEA_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.12 SPEB watchpoint control 1 register 1 (GTMDI_SPEB_WPC1)

The GTMDI_SPEB_WPC1 register shown in *Figure 428* controls the Watchpoint Triggers issued by SPEB channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and to control logic outside of the GTMDI module, such as SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_SPEB_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 35

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	SSEL			0	WMC1	TSS1		0	0	HEN1	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	WMC2	TSS2		0	0	HEN2	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 428. SPEB watchpoint control 1 register (GTMDI_SPEB_WPC1)

Table 491 describes the GTMDI_SPEB_WPC1 register functions.

Table 491. GTMDI_SPEB_WPC1 field descriptions

Field	Description
26–24 SSEL	SPEB sub-module Source Selection(1). Selects which SPE sub-module is the source for watchpoint generation. 000 SPE1 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEB NIPD to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
21–20 TSS1	SPEB NIPD Slope Selection. The TSS1 field selects the slope for the NIPD signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	Halt Enable 1. The HEN1 Halt Enable 1bit controls if the SPEB NIPD active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEB DIR output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0Watchpoint Messages disabled 1 Watchpoint Messages enabled
5–4 TSS2	SPEB DIR Slope Selection. The TSS2 field selects the slope for the DIR signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
1 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the SPEB DIR active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM

1. The SSEL field for SPEB starts from SPE1 selection, differently from the equivalent field for SPEA which selects SPE0 instead.

42.5.1.13 SPEB watchpoint control 2 register (GTMDI_SPEB_WPC2)

The GTMDI_SPEB_WPC2 register shown in *Figure 429* controls the Watchpoint Triggers issued by SPEB. These watchpoints are used for GTM to enter Halt state, generate

watchpoint trace messages and to control logic outside the GTMDI module, such as SPU interface.

Register index: 36

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 429. SPEB watchpoint control 2 register (GTMDI_SPEB_WPC2)

Table 492 describes the GTMDI_SPEB_WPC2 register functions.

Table 492. GTMDI_SPEB_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available watchpoint triggers which one is connected to the NIPD SPEB signal. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the SPEB NIPD selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the SPEB NIPD selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_SPEB_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 492. GTMDI_SPEB_WPC2 field descriptions(Continued)

Field	Description
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DIR SPEB signal. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the SPEB DIR selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the SPEB DIR selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_SPEB_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.14 DPLL watchpoint control 1 register (GTMDI_DPLL_WPC1)

The GTMDI_DPLL_WPC1 register shown in *Figure 430* controls various trace and debug features of the DPLL modules as described below.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_DPLL_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 41

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	TSEL1		0	WMC1	TSS1		0	0	HEN1	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSEL2B		0	0	TSEL2A		0	WMC2	0	0	0	0	HEN2	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 430. DPLL watchpoint control 1 register (GTMDI_DPLL_WPC1)

Table 493 describes the GTMDI_DPLL_WPC1 register functions.

Table 493. GTMDI_DPLL_WPC1 field descriptions

Field	Description
25–24 TSEL1	Watchpoint Selection 1. The TSEL1[1:0] field selects if a watchpoint trigger is issued based on TASI or SASI events. 00 Watchpoint on TASI event 01 Watchpoint on SASI event 10 Reserved 11 Reserved
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on DPLL SASI or TASI to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
21–20 TSS1	TASI/SASI Slope Selection. The TSS1 field selects the slope for TASI or SASI depending on TSEL1 selection. The selected slope is monitored by watchpoints. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	The HEN1 Halt Enable 1 bit controls if the DPLL SASI or TASI selected signal is enabled to HALT the GTM module. If this bit is set and an event occurs on the DPLL selected signal, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
13–12 TSEL2B	Watchpoint Selection 2B. The TSEL2B field selects the RAM access type to mach on 00 Watchpoint is issued on any access (read or write) 01 Watchpoint is issued based on write accesses 10 Watchpoint is issued based on read accesses 11 Reserved

Table 493. GTMDI_DPLL_WPC1 field descriptions(Continued)

Field	Description
9–8 TSEL2A	Watchpoint Selection 2A. The TSEL2A field selects if a watchpoint is issued based on the RAM1a, RAM1bc or RAM2 access. 00 Watchpoint is issued based on RAM1a access 01 Watchpoint is issued based on RAM1b access 10 Watchpoint is issued based on RAM2 access 11 Reserved
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables selected accesses on DPLL RAMs to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
1 HEN2	Halt Enable 2. The HEN Halt Enable bit controls if watchpoint from TSEL2 selection is enabled to HALT the GTM module. If this bit is set and an event occurs on an DPLL selected RAM, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM

42.5.1.15 DPLL watchpoint control 2 register (GTMDI_DPLL_WPC2)

The GTMDI_DPLL_WPC2 register shown in *Figure 431* controls the Watchpoint Triggers issued by DPLL. These triggers are used for GTM to enter Halt state and by GTMDI external logic at SoC level. This register also controls the Watchpoint Trace messages WPM.

Register index: 42

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 431. DPLL watchpoint control 2 register (GTMDI_DPLL_WPC2)

Table 494 describes the GTMDI_DPLL_WPC2 register functions.

Table 494. GTMDI_DPLL_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the DPLL SASI or TASI signal. Other modules can select this same WTSEL1 value, in this case the result trigger signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages related to DPLL SASI or TASI signals. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the DPLL TASI or SASI selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the DPLL TASI or SASI selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_DPLL_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DPLL RAM access signal. Other modules can select this same WTSEL2 value, in this case the result trigger signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used by the watchpoint trace messages related to DPLL RAM access events for RAM1A, RAM1B and RAM2. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 494. GTMDI_DPLL_WPC2 field descriptions(Continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the DPLL RAM selected access. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the DPLL RAM selected access event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_DPLL_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.16 DPLL watchpoint control 3 register (GTMDI_DPLL_WPC3)

The GTMDI_DPLL_WPC3 register shown in *Figure 432* defines the data value observed in the selected DPLL RAM access. A match to this value causes a Watchpoint.

Register index: 43

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DMASK	0	0	0	0	RAM_DATA[23:16]							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RAM_DATA[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 432. DPLL watchpoint control 3 register (GTMDI_DPLL_WPC3)

Table 495 describes the GTMDI_DPLL_WPC3 register functions.

Table 495. GTMDI_DPLL_WPC3 field descriptions

Field	Description
28 DMASK	Data Mask. The DMASK field is a mask for the data comparator. If set, the data read or written to the memory should not be considered for watchpoint generation. 0 RAM_DATA should be compared with data from RAM access 1 RAM_DATA is masked and should not be compared
23–0 RAM_DATA	Data value. The RAM_DATA[23:0] field defines the data value that is compared against the data accessed from the selected DPLL RAM.

42.5.1.17 DPLL watchpoint control 4 register (GTMDI_DPLL_WPC4)

The GTMDI_DPLL_WPC4 register shown in *Figure 433* defines the RAM address mask for the DPLL selected RAM address comparison.

Register index: 44

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RAM_ADDR_MASK											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 433. DPLL watchpoint control 4 register (GTMDI_DPLL_WPC4)

Table 496 describes the GTMDI_DPLL_WPC4 register functions.

Table 496. GTMDI_DPLL_WPC4 field descriptions

Field	Description
11–0 RAM_ADDR_MASK	RAM Address mask. The RAM_ADDR_MASK[11:0] field is a mask applied bit by bit to the RAM_ADDR defined in the GTMDI_DPLL_WPC5 register. If mask is 1, the corresponding RAM_ADDR bit is not valid for comparison, thus it always returns a match when compared to the selected RAM address.

42.5.1.18 DPLL watchpoint control 5 register (GTMDI_DPLL_WPC5)

The GTMDI_DPLL_WPC5 register shown in *Figure 434* defines the address value that is monitored during DPLL selected RAM access. Once this address value matches with the DPLL RAM address a Watchpoint is issued.

Register index: 45

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RAM_ADDR											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 434. DPLL watchpoint control 5 register (GTMDI_DPLL_WPC5)

Table 497 describes the GTMDI_DPLL_WPC5 register functions.

Table 497. GTMDI_DPLL_WPC5 field descriptions

Field	Description
11–0 RAM_ADDR	RAM Address Compare. The RAM_ADDR[11:0] field defines the address value that is compared against the address for write and read DPLL RAM accesses. Before the comparison is executed the RAM_ADDR_MASK should be considered thus masking corresponding bits. A masked bit is always considered a match. **Note:** DPLL module has three RAM modules with different address widths. The address value in the RAM_ADDR field and the RAM address should be right aligned in order to be compared.

42.5.1.19 DPLL data trace control register (GTMDI_DPLL_DTC)

The GTMDI_DPLL_DTC register shown in *Figure 435* controls data trace message transmission.

Register index: 46

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	RWC		0	0	RAMSEL		0	DMC	STSEL		0	0	0	SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 435. DPLL data trace control register (GTMDI_DPLL_DTC)

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the DMC control bit thus controlling the Data Trace messages behavior. In case of JTAG modifies the DMC bit and Start/Stop signals occur at the same time also modifying the same bit, Start/Stop signals have precedence over JTAG writes. The SEN control bit from the GTMDI_DPLL_DTC register need to be set in order to allow external Start/Stop signals to control the DMC bit.

Table 498 describes the GTMDI_DPLL_DTC register functions.

Table 498. GTMDI_DPLL_DTC field descriptions

Field	Description
13–12 RWC	Read/Write Control. This bit controls DPLL data trace message transmission. 00 Transmits only memory reads 01 Transmits only memory writes 10 Transmits all memory accesses 11 Reserved
9–8 RAMSEL	RAM Module Selection. This field selects a DPLL memory module for data tracing. 00 RAM1a selected for data tracing 01 RAM1b selected for data tracing 10 RAM2 selected for data tracing 11 Reserved

Table 498. GTMDI_DPLL_DTC field descriptions(Continued)

Field	Description
6 DMC	Data Trace Message Control. This bit is written by JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Data Trace Messages disabled 1 Data Trace Messages enabled
5–4 STSEL	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
0 SEN	Start/Stop Enable. The SEN bit field enables the DPLL data trace message generator to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DMC Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.20 ARU watchpoint control 1 register (GTMDI_ARU_WPC1)

The GTMDI_ARU_WPC1 register shown in *Figure 436* controls the Watchpoint Triggers issued by ARU Debugging Channels. These watchpoint triggers are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_ARU_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 50

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	TSS1	0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	TSS2	0	0	0	0	HEN2	WMC2	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 436. ARU watchpoint control 1 register (GTMDI_ARU_WPC1)

Table 499 describes the GTMDI_ARU_WPC1 register functions.

Table 499. GTMDI_ARU_WPC1 field descriptions

Field	Description
28 TSS1	ARU Debugging Channel 0 Data Activity Selection 1. The TSS1 bit selects the type of data activity to monitor on ARU Debugging Channel 0. 0 Any Data (do not compare data). Watchpoint generation is based on a valid data indication on ARU Debugging Channel 0 1 Compare data with expected value
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the ARU Debugging Channel 0 is enabled to HALT the GTM module. If this bit is set and an event occurs on the ARU Debugging Channel 0, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches on the ARU Debugging Channel 0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
12 TSS2	ARU Debugging Channel 1 Data Activity Selection. The TSS2 bit selects the type of data activity to monitor on the ARU Debugging Channel 1. 0 Any Data (do not compare data). Watchpoint generation is based on a valid data indication on ARU Debugging Channel Monitor 1 1 Compare data with expected value
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the ARU debugging Channel 1 is enabled to HALT the GTM module. If this bit is set and an event occurs on the ARU Debugging Channel 1, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match on the ARU Debugging Channel 1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

42.5.1.21 ARU watchpoint control 2 register (GTMDI_ARU_WPC2)

The GTMDI_ARU_WPC2 register shown in *Figure 436* controls the Watchpoint Triggers issued by ARU Debugging Channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 51

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 437. ARU watchpoint control 2 register (GTMDI_ARU_WPC2)

Table 500 describes the GTMDI_ARU_WPC2 register functions.

Table 500. GTMDI_ARU_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the ARU Debugging Channel 0. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages for ARU Debugging Channel 0. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the ARU debugging Channel 0. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the ARU Debugging Channel 0 to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_ARU_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 500. GTMDI_ARU_WPC2 field descriptions(Continued)

Field	Description
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the ARU Debugging Channel 1. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control for ARU Debugging Channel 1. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the ARU Debugging Channel 1 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the ARU Debugging Channel 1 to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_ARU_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.22 ARU watchpoint DATA0H register (GTMDI_ARU_DATA0H)

The ARU Watchpoint DATA0H high word shown in *Figure 438* is used for comparison against the high word of the ARU Debugging Channel 0. The address of the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS0 register.

Register index: 52

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA0H[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA0H[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 438. ARU watchpoint DATA0H register (GTMDI_ARU_DATA0H)

Table 501 describes the GTMDI_ARU_DATA0H register functions.

Table 501. GTMDI_ARU_DATA0H field descriptions

Field	Description
28–0 DATA0H	Watchpoint DATA0H. This field is compared against the high word of the ARU Debugging Channel 0.

42.5.1.23 ARU watchpoint DATA0L register (GTMDI_ARU_DATA0L)

The ARU Watchpoint DATA0H low word shown in *Figure 439* is used for comparison against the low word of the ARU Debugging Channel 0. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS0 register.

Register index: 53

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA0L[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA0L[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 439. ARU watchpoint DATA0L register (GTMDI_ARU_DATA0L)

Table 502 describes the GTMDI_ARU_DATA0L register functions.

Table 502. GTMDI_ARU_DATA0L field descriptions

Field	Description
28–0 DATA0L	Watchpoint DATA0L. This field is compared against the low word of the ARU Debugging Channel 0.

42.5.1.24 ARU watchpoint DATA1H register (GTMDI_ARU_DATA1H)

The ARU Watchpoint DATA1H high word shown in *Figure 440*. is used for comparison against the high word of the ARU Debugging Channel 1. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS1 register.

Register index: 54

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA1H[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA1H[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 440. ARU watchpoint DATA1H register (GTMDI_ARU_DATA1H)

Table 503 describes the GTMDI_ARU_DATA1H register functions.

Table 503. GTMDI_ARU_DATA1H field descriptions

Field	Description
28–0 DATA1H	Watchpoint DATA1H. This field is compared against the high word of the ARU Debugging Channel 1.

42.5.1.25 ARU watchpoint DATA1L register (GTMDI_ARU_DATA1L)

The ARU Watchpoint DATA1L low word shown in *Figure 441* is used to be compared against the low word of the ARU Debugging Channel 1. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS1 register.

Register index: 55

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA1L[28:16]												
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA1L[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 441. ARU Watchpoint DATA1L (GTMDI_ARU_DATA1L)

Table 504 describes the GTMDI_ARU_DATA1L register functions.

Table 504. GTMDI_ARU_DATA1L field descriptions

Field	Description
28–0 DATA1L	Watchpoint DATA1L. This field is compared against the low word of the ARU Debugging Channel 1.

42.5.1.26 ARU data trace control register (GTMDI_ARU_DTC)

The GTMDI_ARU_DTC register shown in *Figure 442* controls the ARU data trace message transmission. The GTM Module selects two ARU addresses to be monitored, using the ARU Debugging Channels 0 and 1. The data trace for these two channels is controlled by DMC1/2 bits in this register. Once enabled, messages are generated as soon as a valid data is identified in one of the ARU Debugging Channels, 0 or 1.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the DMC1 and DMC2 control bits thus controlling the Data Trace messages behavior. In case of JTAG modifies the DMC1 or DMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits need to be set in order to allow external Start/Stop signals to control the DMC1/2 bits.

Register index: 56

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	STSEL1		0	0	DMC1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	STSEL2		0	0	DMC2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 442. ARU data trace control register (GTMDI_ARU_DTC)

Table 505 describes the GTMDI_ARU_DTC register functions.

Table 505. GTMDI_ARU_DTC field descriptions

Field	Description
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the ARU Debugging Channel 0 data trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 DMC1	Data Trace Message Control. This bit controls ARU data trace messages transmission for ARU Debugging Channel 0. This bit is written by the JTAG interface but can also be controlled at SoC level by dedicated Start/Stop signals, inputs to GTMDI. 0 Data Trace Messages disabled 1 Data Trace Messages enabled
16 SEN1	Start/Stop Enable 1. The SEN 1 bit field enables the ARU data trace message generator for ARU Debugging Channel 0 to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DMC1 Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used by the ARU Debugging Channel 1 data trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 505. GTMDI_ARU_DTC field descriptions(Continued)

Field	Description
1 DMC2	Data Trace Message Control. This bit controls ARU data trace message transmission for ARU Debugging Channel 1. This bit is written by the JTAG interface but can also be controlled at SoC level by dedicated Start/Stop signals, inputs to GTMDI. 0 Data Trace Messages disabled 1 Data Trace Messages enabled
0 SEN2	The SEN2 bit field enables the ARU data trace message generator for ARU Debugging Channel 1 to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to DMC2 Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.27 MCSA/B development control register (GTMDI_MCSA_DC and GTMDI_MCSB_DC)

The GTMDI_MCSA_DC and GTMDI_MCSB_DC registers shown in *Figure 443* control various trace and debug features.

Note: Since MCSA and MCSB have a common sets of registers they are described together to avoiding duplication of similar information.

Register index: MCSA–60 / MCSB–77

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	MCSA/B_SEL				0	0	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 443. MCSA/B development control register (GTMDI_MCSA_DC and GTMDI_MCSB_DC)

Table 506 describes the GTMDI_MCSA_DC and GTMDI_MCSB_DC register functions.

Table 506. GTMDI_MCSA_DC and GTMDI_MCSB_DC field descriptions

Field	Description
11–8 MCSA/B_SEL	MCSA/B Selection field. The MCSA/B_SEL[3:0] selection field selects which one of the MCS cores in the GTM module are assigned as MCSA/B. 0000 MCS0 0001 MCS1 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

42.5.1.28 MCSA/B watchpoint control register (GTMDI_MCSA_WPC and GTMDI_MCSB_WPC)

The MCSA/B Watchpoint Control Registers shown in *Figure 444* are used to configure MCSA/B watchpoints. These registers can be configured for halt/watchpoint based on instruction fetch, data read or data write, with address and data masking options.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC control bit thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC bit and Start/Stop signals occur at the same time also modifying the same bit, Start/Stop signals have precedence over JTAG writes. The SEN control bit needs to be asserted in order to allow external Start/Stop signals to control the WMC bit.

Note: The watchpoint channel selection is made through the GTMDI_MCSA_CE and GTMDI_MCSB_CE by the WPCE field.

Register index: MCSA–62 / MCSB–79

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	HACH		0	WMC	0	0	HME			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HWO		0	0	0	0	0	EOC	HEN	0	STSEL		0	0	TEN	SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 444. MCSA/B watchpoint control register (GTMDI_MCSA_WPC and GTMDI_MCSB_WPC)

Table 507 describes the GTMDI_MCSA_WPC and GTMDI_MCSB_WPC register functions.

Table 507. GTMDI_MCSA_WPC and GTMDI_MCSB_WPC field descriptions

Field	Description
25–24 HACH	Halt/Watchpoint on Active Channel. The HACH field selects which type of MCS RAM access generates a watchpoint or halt. Combining this field with HWO gives GTMDI the capability of monitoring channel activity as well as accesses to specific addresses or data. 00 Match on Instruction Fetch 01 Match on Data Read 10 Match on Data Write 11 Match on Data Read or Write
22 WMC	Watchpoint Message Control. The WMC Watchpoint Message Control bit controls the watchpoint message sent through the Message Data bus. If this bit is asserted it enables MCSA/B Watchpoints to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
19–16 HME	Halt/Watchpoint Data Mask Enable. The HME field selects which data bytes are used for hardware halt/watchpoint generation when comparing to HWD register. This register field is only meaningful if HWO bit field is [x1], otherwise the data comparison is disabled independent of HME value. 0000 All bytes of data compared xxx1 Least significant byte of data masked out xx1x Second least significant byte of data masked out x1xx Second most significant byte of data masked out 1xxx Most significant byte of data masked out
15–14 HWO	Halt/Watchpoint Operand. The HWO field selects if address and/or data matching is done. Registers WPA and WPD are used for address and data comparison, respectively. If address is not compared WPA is not use. If data is not compared then WPD is not used. 00 Comparison Disabled for address or data x1 Compare data with HWD field 1x Compare address with HWA field
8 EOC	$\overline{\text{EVTO}}$ Control. The EOC bit controls the indication on the $\overline{\text{EVTO}}$ output. See *Figure 447*, *Figure 448*, *Figure 449* and *Figure 450* for definition about watchpoint 1 and 2. 0 Indicates Watchpoint 2 occurrence 1 Indicates Watchpoint 1 occurrence
7 HEN	Halt Enable. The HEN Halt Enable bit controls if the MCSA/B Watchpoint is enabled to HALT the GTM IP module. If this bit is set and watchpoint occurs on the MCSA/B a Halt signal is generated to the GTM IP which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
5–4 STSEL	Start/Stop input signal selection. Selects the Start/Stop input that is used to control the watchpoint messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 507. GTMDI_MCSA_WPC and GTMDI_MCSB_WPC field descriptions(Continued)

Field	Description
1 TEN	Watchpoint Trigger Enable.The TEN field controls if a external trigger is issued by the MCSA/B when a watchpoint occurs. 0 Disables Trigger Generation 1 Enables Trigger Generation
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA/B Watchpoint message to consider the Start/Stop inputs for watchpoint message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC Watchpoint Trace Message Control bit. 0 Disables trace control from Start/Stop inputs 1 Enables trace control from Start/Stop inputs

42.5.1.29 MCSA/B program fetch trace control register (GTMDI_MCSA_PTC and GTMDI_MCSB_PTC)

The GTMDI_MCSA_PTC and GTMDI_MCSB_PTC registers shown in *Figure 445* allow program fetch traces to be enabled and/or disabled on the occurrence of a watchpoint. If the same watchpoint is programmed to enable and disable a trace, the occurrence of the watchpoint toggles the current value of the trace flags. If one watchpoint occurs to enable trace and a different watchpoint occurs to disable trace at the same time the current value of the trace flags also toggle.

Register index: MCSA–63 / MCSB–80

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PFTS		0	0	PFTE		0	FTC	0	0	0	0	0	SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 445. MCSA/B program fetch trace control register (GTMDI_MCSA_PTC and GTMDI_MCSB_PTC)

Table 508 describes the GTMDI_MCSA_PTC and GTMDI_MCSB_PTC register functions.

Table 508. GTMDI_MCSA_PTC and GTMDI_MCSB_PTC field descriptions

Field	Description
13–12 PFTS	Program Fetch Trace Start. This bit field allows Program fetch trace to be enabled at a watchpoint occurrence. See *Figure 447*, *Figure 448*, *Figure 449* and *Figure 450* for definition about watchpoint 1 and 2. 00 No watchpoint selected (do not start Program Fetch Trace) 01 Program Fetch Trace enabled on MCSA/B Program Watchpoint 1 occurrence 10 Program Fetch Trace enabled on MCSA/B Program Watchpoint 2 occurrence 11 Reserved
9–8 PFTE	Program Fetch Trace End. This bit field allows program trace to be disabled at a watchpoint occurrence. 00 No watchpoint selected 01 Program Fetch Trace disabled on MCSA/B Program Watchpoint 1 occurrence 10 Program Fetch Trace disabled on MCSA/B Program Watchpoint 2 occurrence 11 Reserved
6 FTC	Fetch Trace Control. This bit controls the Fetch Trace message sent through the Message Data bus. If this bit is asserted it enables MCSA/B Fetch access to generate a trace message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI 0 Fetch Messages disabled 1 Fetch Messages enabled
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA/B Fetch message to consider the Start/Stop inputs for message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the FTC Fetch Trace Message Control bit. 0 Disable fetch trace control from Start/Stop inputs 1 Enable fetch trace control from Start/Stop inputs

42.5.1.30 MCSA/B data trace control register (GTMDI_MCSA_DTC and GTMDI_MCSB_DTC)

The GTMDI_MCSA_DTC and GTMDI_MCSB_DTC registers shown in *Figure 446* control which address ranges are enabled for data trace, and if reads and/or writes are traced in that range.

Register index: MCSA–64 / MCSB–81

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RC0	RC1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	RWT0		0	0	RWT1		0	DTC	DTS		0	DTE		SEN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 446. MCSA/B data trace control register (GTMDI_MCSA_DTC and GTMDI_MCSB_DTC)

Table 509 describes the GTMDI_MCSA_DTC and GTMDI_MCSB_DTC register functions.

Table 509. GTMDI_MCSA_DTC and GTMDI_MCSB_DTC field descriptions

Field	Description
17 RC0	Range Control 0. Controls the range on data trace window 0. The window 0 is defined by DTAR1 register. See *Figure 452*. 0 Trace addresses outside (exclusive) of data trace window 0 1 Trace address inside (inclusive) of data trace window 0
16 RC1	Range Control 1. Controls the range on data trace window 1. The window 1 is defined by DTAR2 register. See *Figure 453*. 0 Trace addresses outside (exclusive) of data trace window 1 1 Trace address inside (inclusive) of data trace window 1
13–12 RWT0	MCSA/B Read/Write Trace Control 0. RWT0 controls whether data trace messages are generated for MCSA/B RAM read or write accesses. 00 No MCSA/B data trace messages generated for data trace window 0 01 Enable MCSA/B data read trace for MCS window 0 10 Enable MCSA/B data write trace for MCS window 0 11 Enable MCSA/B data read and write trace for data trace window 0
9–8 RWT1	MCSA/B Read/Write Trace Control 1. RWT1 controls whether data trace messages are generated for MCSA/B accesses inside MCS data trace window, and if so, whether reads, writes, or both generate the data trace messages. 00 No MCSA/B data trace messages generated for data trace window 1 01 Enable MCSA/B data read trace for MCS window 1 10 Enable MCSA/B data write trace for MCS window 1 11 Enable MCSA/B data read and write trace for data trace window 1
6 DTC	Data Trace Control. This bit controls the Data Trace message sent through the Message Data bus. If this bit is asserted it enables MCSA/B Data access to generate a data trace message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI 0 Data Messages disabled 1Data Messages enabled
5–4 DTS	Data Trace Start. This bit field allows data trace to be enabled at a watchpoint occurrence. 00 No watchpoint selected 01 Data Trace enabled on MCSA/B Watchpoint 1 occurrence 10 Data Trace enabled on MCSA/B Watchpoint 2 occurrence 11 Reserved
1–0 DTE	Data Trace End. This bit field allows data trace to be disabled at a watchpoint occurrence. 00 No watchpoint selected 01 Data Trace disabled on MCSA/B Watchpoint 1 occurrence 10 Data Trace disabled on MCSA/B Watchpoint 2 occurrence 11 Reserved
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA/B Data message to consider the Start/Stop inputs for message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DTC Data Trace Message Control bit. 0 Disable data trace control from Start/Stop inputs 1 Enable data trace control from Start/Stop inputs

42.5.1.31 MCSA/B watchpoint address 1 register (GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1)

The MCSA/B Watchpoint Address Registers 1 shown in *Figure 447* are used to configure the MCSA/B halt/watchpoint hardware. They are used for data and instruction fetch accesses.

Register index: MCSA–65 / MCSB–82

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	HWAM													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	HWA													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 447. MCSA/B watchpoint address 1 register (GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1)

Note: The HWA field represents word addresses relative to the MCS address map, not the CPU address map. This applies to GTMDI_MCSA_WPA1, GTMDI_MCSB_WPA1, GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2 registers.

Table 510 describes the GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1 register functions.

Table 510. GTMDI_MCSA_WPA1 and GTMDI_MCSB_WPA1 field descriptions

Field	Description
29–16 HWAM	Halt/Watchpoint Address Mask. The HWAM field is used to select which address bits are compared for halt and watchpoint generation. Address match = (HWA&HWAM) == (Address & HWAM);
13–0 HWA	Halt/Watchpoint Address. The HWA field is used for comparison with instruction or data addresses. The HWA field represents a word address.

42.5.1.32 MCSA/B watchpoint address 2 register (GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2)

The MCSA/B Watchpoint Address 2 Registers shown in *Figure 448* are used to configure the MCSA/B halt/watchpoint hardware. They are used for data and instruction fetch accesses.

Register index: MCSA–66 / MCSB–83

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	HWAM													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	HWA													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 448. MCSA/B watchpoint address 2 register (GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2)

Table 511 describes the GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2 register functions.

Table 511. GTMDI_MCSA_WPA2 and GTMDI_MCSB_WPA2 field descriptions

Field	Description
29–16 HWAM	Halt/Watchpoint Address Mask. The HWAM field is used to select which address bits are compared for halt and watchpoint generation. Address match = (HWA&HWAM) == (Address & HWAM);
13–0 HWA	Halt/Watchpoint Address. The HWA field is used for comparison with instruction or data addresses. The HWA field represents a word address.

42.5.1.33 MCSA/B watchpoint data 1 register (GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1)

The MCSA/B Halt/Watchpoint Data 1 Registers shown in *Figure 449* are used to configure the MCSA/B halt/watchpoint hardware. They are used for data accesses selection only.

Register index: MCSA–67 / MCSB–84

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HWD															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HWD															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 449. MCSA/B watchpoint data 1 register (GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1)

Table 512 describes the GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1 register functions.

Table 512. GTMDI_MCSA_WPD1 and GTMDI_MCSB_WPD1 field descriptions

Field	Description
31–0 HWD	Halt/Watchpoint Data. The HWD field is used to compare data operands in a RAM read or write access. Depending upon the size of the access, only certain bits of HWD are compared against the RAM data. If the operation size is 8 bits, the 8 most significant bits of HWD are compared against the RAM data. If the operation size is 24 bits, the 24 least significant bits of HWD are compared against the RAM data. If the operation size is 32 bits, all bits from HWD are compared against the RAM data.

42.5.1.34 MCSA/B watchpoint data 2 register (GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2)

The MCSA/B Watchpoint Data 2 Registers shown in *Figure 450* are used to configure the MCSA/B halt/watchpoint hardware. They are used for data accesses selection only.

Register index: MCSA–68 / MCSB–85

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HWD															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HWD															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 450. MCSA/B watchpoint data 2 register (GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2)

Table 513 describes the GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2 register functions.

Table 513. GTMDI_MCSA_WPD2 and GTMDI_MCSB_WPD2 field descriptions

Field	Description
31–0 HWD	Halt/Watchpoint Data. The HWD field is used to compare data operands in a RAM read or write access. Depending upon the size of the access, only certain bits of HWD are compared against the RAM data. If the operation size is 8 bits, the 8 most significant bits of HWD are compared against the RAM data. If the operation size is 24 bits, the 24 least significant bits of HWD are compared against the RAM data. If the operation size is 32 bits, all bits from HWD are compared against the RAM data.

42.5.1.35 MCSA/B channel enable register (GTMDI_MCSA_CE and GTMDI_MCSB_CE)

The GTMDI_MCSA_CE and GTMDI_MCSB_CE registers shown in *Figure 451* enable the monitoring for 8 MCSA/B channels independently.

Register index: MCSA–69 / MCSB–86

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CFTE								CDTE							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	WPCE							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 451. MCSA/B channel enable register (GTMDI_MCSA_CE and GTMDI_MCSB_CE)

Table 514 describes the GTMDI_MCSA_CE and GTMDI_MCSB_CE register functions.

Table 514. GTMDI_MCSA_CE and GTMDI_MCSB_CE field descriptions

Field	Description
7–0 CFTE	MCSA/B Channel Fetch Trace Enable. The CFTE[7:0] is an one hot bit field which defines the channel or channels to be monitored for fetch traces. If enabled, the corresponding channel Fetches are monitored by Fetch Trace messages. Start and Stop message controls are controlled by the FTC bit in the GTMDI_MCSA/B_PTC registers (*Section 42.5.1.29, MCSA/B program fetch trace control register (GTMDI_MCSA_PTC and GTMDI_MCSB_PTC)*. 00000000No channel selected for trace xxxxxxx1Channel [0] selected xxxxxx1xChannel [1] selected xxxxx1xxChannel [2] selected xxxx1xxxChannel [3] selected xxx1xxxxChannel [4] selected xx1xxxxxChannel [5] selected x1xxxxxxChannel [6] selected 1xxxxxxxChannel [7] selected 11111111All Channels selected

Table 514. GTMDI_MCSA_CE and GTMDI_MCSB_CE field descriptions(Continued)

Field	Description
7–0 CDTE	MCSA/B Channel Data Trace Enable. The CDTE[7:0] is an one hot bit field which defines the channel or channels to be monitored for data trace messages. If enabled, the corresponding channel Data Accesses are monitored by Data Trace messages. Start and Stop message are controlled by the DTC bit in the GTMDI_MCSA/B_DTC registers (*Section 42.5.1.30, MCSA/B data trace control register (GTMDI_MCSA_DTC and GTMDI_MCSB_DTC)* and by the Data trace window. 00000000No channel selected for trace xxxxxxx1Channel [0] selected xxxxxx1xChannel [1] selected xxxxx1xxChannel [2] selected xxxx1xxxChannel [3] selected xxx1xxxxChannel [4] selected xx1xxxxxChannel [5] selected x1xxxxxxChannel [6] selected 1xxxxxxxChannel [7] selected 11111111All Channels selected
7–0 WPCE	MCSA/B Channel Enable. The WPCE[7:0] is a one hot bit field which defines the channel or channels to be monitored for watchpoint, data and fetch traces. The messages controlled by this bit must be also enabled and disabled according to the occurrence of Watchpoints and Address Ranges. 00000000No channel selected for trace xxxxxxx1Channel [0] selected xxxxxx1xChannel [1] selected xxxxx1xxChannel [2] selected xxxx1xxxChannel [3] selected xxx1xxxxChannel [4] selected xx1xxxxxChannel [5] selected x1xxxxxxChannel [6] selected 1xxxxxxxChannel [7] selected 11111111All Channels selected

42.5.1.36 MCSA/B data trace address range 1 register (GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1)

The GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1 registers shown in *Figure 452* define a data trace address range. The address range is used by the MCSA/B and enables the range for reads and/or writes access to data only. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 0.

Register index: MCSA–71 / MCSB–88

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DTSA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DTEA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 452. Data trace address range 1 register (GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1)

Note: The DTSA and DTEA fields represent word addresses relative to the MCS address map, not the CPU address.

Table 515 describes the GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1 register functions.

Table 515. GTMDI_MCSA_DTAR1 and GTMDI_MCSB_DTAR1 field descriptions

Field	Description
29–18 DTSA	Data Trace Start Address. The DTSA field is the start address for MCSA/B data trace window.
13–2 DTEA	Data Trace End Address. The DTEA field is the end address for MCSA/B data trace window.

42.5.1.37 MCSA/B data trace address range 2 register (GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2)

The GTMDI_MCSA_DTAR2 and GTMDI_MCSA_DTAR2 registers shown in *Figure 453* define a data trace address range. The address range is used by the MCSA/B and enables the range for reads and/or writes access to data only. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 1.

Register index: MCSA–72 / MCSB–89

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DTSA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DTEA [13:2]												0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 453. MCSA data trace address range 2 register (GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2)

Note: DTSA and DTEA fields represent word addresses which are included in the address range.

Table 516 describes the GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR12 register functions.

Table 516. GTMDI_MCSA_DTAR2 and GTMDI_MCSB_DTAR2 field descriptions

Field	Description
29–18 DTSA	Data Trace Start Address. The DTSA field is the start address for MCSA/B data trace window.
13–2 DTEA	Data Trace End Address. The DTEA field is the end address for MCSA/B data trace window.

42.5.1.38 TBU0 watchpoint control 1 register (GTMDI_TBU0_WPC1)

The GTMDI_TBU0_WPC1 register shown in *Figure 454* controls the Watchpoint issued by TBU0 GTM Time Base Units (TBU). These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TBU0_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 94

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	TSS1			0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TSS2			0	0	0	0	HEN2	WMC2	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 454. TBU0 watchpoint control 1 register (GTMDI_TBU0_WPC1)

Table 517 describes the GTMDI_TBU0_WPC1 register functions.

Table 517. GTMDI_TBU0_WPC1 field descriptions

Field	Description
30–28 TSS1	TBU0 Watchpoint 1 Compare Selection. The TSS1[2:0] field controls the type of comparison used for TBU0 watchpoint generation. 000 TBU0 24-bit update (do not compare data) 001 TBU 27-bit update (do not compare data) 010 Compare TBU0 value in 24-bit mode with expected data(1) 011 Compare TBU0 value in 27-bit mode with expected data 100 Reserved 101 Reserved 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU0 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU0 a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
14–12 TSS2	TBU0 Watchpoint 2 Compare Selection 2. The TSS2[2:0] field controls the type of comparison used for TBU0 watchpoint generation. 000 TBU0 24-bit update (do not compare data) 001 TBU 27-bit update (do not compare data) 010 Compare TBU0 value in 24-bit mode with expected data(1) 011 Compare TBU0 value in 27-bit mode with expected data 100 Reserved 101 Reserved 110 Reserved 111 Reserved
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU0 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU0, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0Watchpoint Messages disabled 1 Watchpoint Messages enabled

1. Only bits from 26 to 3 from register GTMDI_TBU0_DATA are used for comparison

42.5.1.39 TBU0 watchpoint control 2 register (GTMDI_TBU0_WPC2)

The GTMDI_TBU0_WPC2 shown in *Figure 455* register controls the Watchpoint Triggers issued by TBU0. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 95

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 455. TBU0 watchpoint control 2 register (GTMDI_TBU0_WPC2)

Table 518 describes the GTMDI_TBU0_WPC2 register functions.

Table 518. GTMDI_TBU0_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU0 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1.The TEN1 field controls if a watchpoint external trigger is issued by the TBU0 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU0 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU0_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 518. GTMDI_TBU0_WPC2 field descriptions(Continued)

Field	Description
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU0 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU0 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU0 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU0_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.40 TBU0 watchpoint DATA register (GTMDI_TBU0_DATA)

The TBU0 Watchpoint DATA shown in *Figure 456* is used for watchpoints generation for matching with the TBU0 value.

Register index: 96

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	TBU0_DATA[26:16]										
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU0_DATA[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 456. TBU0 watchpoint DATA register (GTMDI_TBU0_DATA)

Table 519 describes the GTMDI_TBU0_DATA register functions.

Table 519. GTMDI_TBU0_DATA field descriptions

Field	Description
26–0 TBU0_DATA	TBU0 Watchpoint DATA. The field TBU0_DATA is used to compare against the TBU0 data.

42.5.1.41 TBU1 watchpoint control 1 register (GTMDI_TBU1_WPC1)

The GTMDI_TBU1_WPC1 register shown in *Figure 457* controls the Watchpoint issued by TBU1 GTM Time Base Units. These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TBU1_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 101

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	0	0	0	HEN2	WMC2	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 457. TBU1 watchpoint control 1 register (GTMDI_TBU1_WPC1)

Table 520 describes the GTMDI_TBU1_WPC1 register functions.

Table 520. GTMDI_TBU1_WPC1 field descriptions

Field	Description
29–28 TSS1	TBU1 Watchpoint 1 Compare Selection. The TSS1[1:0] field controls the type of comparison used for TBU1 watchpoint generation. 00 TBU1 update (do not compare data) 01 Compare TBU1 value with expected data 10 Reserved 11 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU1 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU1 a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM

Table 520. GTMDI_TBU1_WPC1 field descriptions(Continued)

Field	Description
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
13–12 TSS2	TBU1 Watchpoint 2 Compare Selection 2. The TSS2[1:0] field controls the type of comparison used for TBU1 watchpoint generation. 00 TBU1 update (do not compare data) 01 Compare TBU1 value with expected data 10 Reserved 11 Reserved
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU1 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU1, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0Watchpoint Messages disabled 1 Watchpoint Messages enabled

42.5.1.42 TBU1 watchpoint control 2 register (GTMDI_TBU1_WPC2)

The GTMDI_TBU1_WPC2 register shown in *Figure 458* controls the Watchpoint Triggers issued by TBU1. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 102

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 458. TBU1 watchpoint control 2 register (GTMDI_TBU1_WPC2)

Table 521 describes the GTMDI_TBU1_WPC2 register functions.

Table 521. GTMDI_TBU1_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU1 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1.The TEN1 field controls if a watchpoint external trigger is issued by the TBU1 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU1 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU1_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU1 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 521. GTMDI_TBU1_WPC2 field descriptions(Continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU1 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU1 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU1_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.43 TBU1 watchpoint DATA register (GTMDI_TBU1_DATA)

The TBU1 Watchpoint DATA shown in *Figure 459* is used for watchpoints generation for matching with the TBU1 value.

Register index: 103

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TBU1_DATA[23:16]							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU1_DATA[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 459. TBU1 watchpoint DATA register (GTMDI_TBU1_DATA)

Table 522 describes the GTMDI_TBU1_DATA register functions.

Table 522. GTMDI_TBU1_DATA field descriptions

Field	Description
23–0 TBU1_DATA	TBU1 Watchpoint DATA. The field TBU1_DATA[23:0] is used to compare against the TBU1 data.

42.5.1.44 TBU2 watchpoint control 1 register (GTMDI_TBU2_WPC1)

The GTMDI_TBU2_WPC1 register shown in *Figure 460* controls the Watchpoint issued by TBU2 GTM Time Base Units. These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signals inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2

control bits from the GTMDI_TBU2_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Register index: 108

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	0	0	0	HEN2	WMC2	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 460. TBU2 watchpoint control 1 register (GTMDI_TBU2_WPC1)

Table 523 describes the GTMDI_TBU2_WPC1 register functions.

Table 523. GTMDI_TBU2_WPC1 field descriptions

Field	Description
29–28 TSS1	TBU2 Watchpoint 2 Compare Selection. The TSS1[1:0] field controls the type of comparison used for TBU2 watchpoint generation. 00 TBU2 update (do not compare data) 01 Compare TBU2 value with expected data 10 Reserved 11 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU2 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU2 a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU2 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
13–12 TSS2	TBU2 Watchpoint 2 Compare Selection 2. The TSS2[1:0] field controls the type of comparison used for TBU2 watchpoint generation. 00 TBU2 update (do not compare data) 01 Compare TBU2 value with expected data 10 Reserved 11 Reserved

Table 523. GTMDI_TBU2_WPC1 field descriptions(Continued)

Field	Description
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU2 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU2, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU2 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals inputs to GTMDI. 0Watchpoint Messages disabled 1 Watchpoint Messages enabled

42.5.1.45 TBU2 watchpoint control 2 register (GTMDI_TBU2_WPC2)

The GTMDI_TBU2_WPC2 register shown in *Figure 461* controls the Watchpoint Triggers issued by TBU2. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Register index: 109

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 461. TBU2 watchpoint control 2 register (GTMDI_TBU2_WPC2)

Table 524 describes the GTMDI_TBU2_WPC2 register functions.

Table 524. GTMDI_TBU2_WPC2 field descriptions

Field	Description
30–28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU2 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21–20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1.The TEN1 field controls if a watchpoint external trigger is issued by the TBU2 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU2 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU2_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14–12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU2 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5–4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 524. GTMDI_TBU2_WPC2 field descriptions(Continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU2 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU2 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU2_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

42.5.1.46 TBU2 watchpoint DATA register (GTMDI_TBU2_DATA)

The TBU2 Watchpoint DATA shown in *Figure 462* is used for watchpoints generation for matching with the TBU2 value.

Register index: 110

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TBU2_DATA[23:16]							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU2_DATA[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 462. TBU2 watchpoint DATA register (GTMDI_TBU2_DATA)

Table 525 describes the GTMDI_TBU2_DATA register functions.

Table 525. GTMDI_TBU2_DATA field descriptions

Field	Description
23–0 TBU2_DATA	TBU2 Watchpoint DATA. The field TBU2_DATA[23:0] is used to compare against the TBU2 data.

42.5.2 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in *Table 479*. Unimplemented registers should not be accessed.

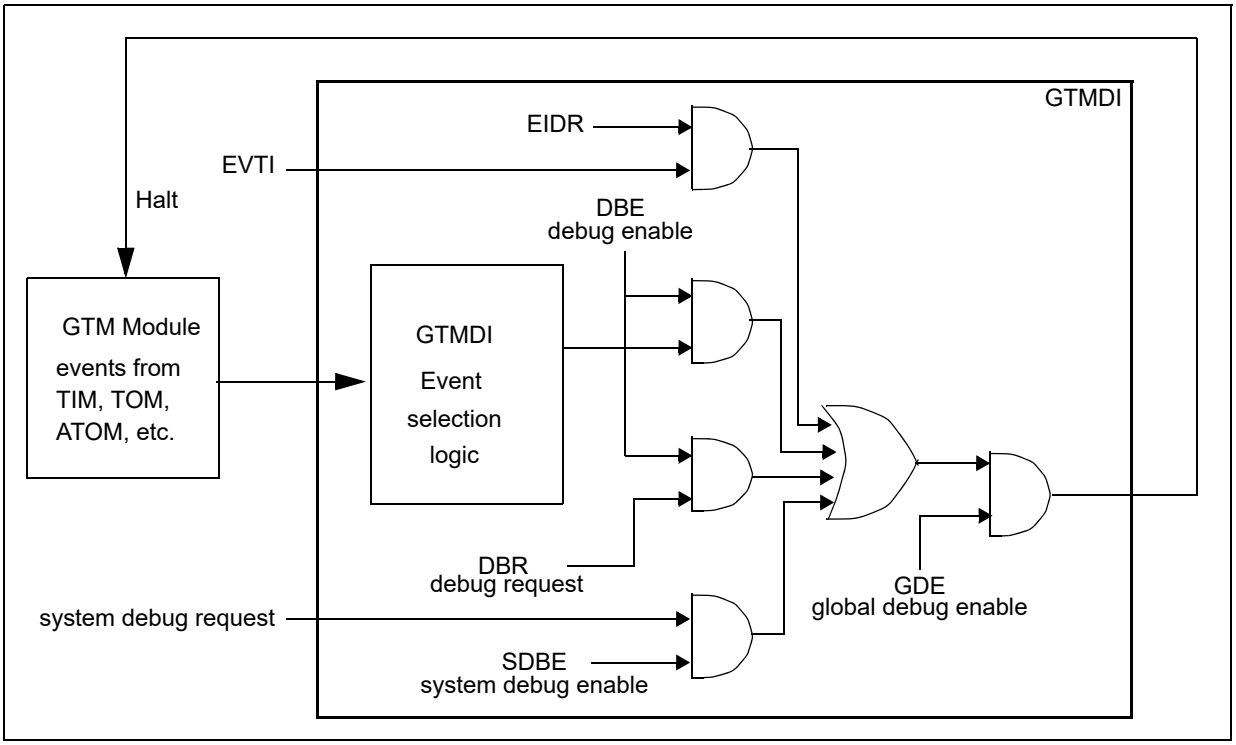
42.6 Functional description

This section describes in detail the main functionality of the GTMDI module.

42.6.1 Debug enable logic

In order to enable the GTM to enter Halt state, which is the module debug state, there are some registers that need to be set. The sources that can put GTM in halt state are events from TIM, TOM, ATOM, MCS and several other internal GTM modules that are selected and controlled by the GTMDI_DC register bits shown in *Figure 418*. The system debug request external signal is also a source of halts, as well as direct JTAG write to control register. *Figure 463* describes the logic behind the Halt signal generated by GTMDI. See *Figure 418* for a description of the register bits mentioned in *Figure 463*.

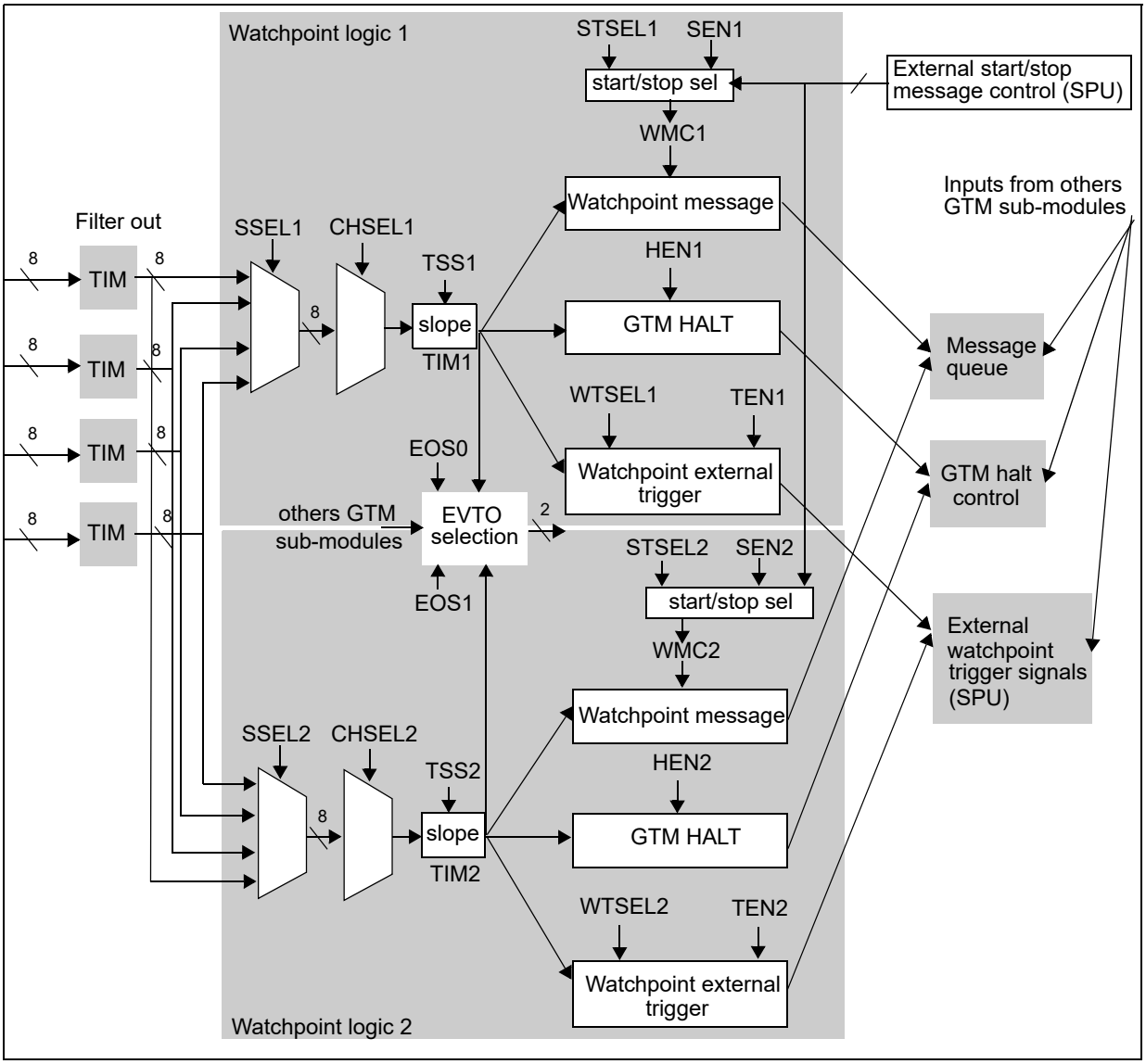
Figure 463. GTM Halt signal enable logic



42.6.2 TIM-TOM-ATOM selection and control logic

The TIM sub-modules can be selected to generate signals that control the watchpoint messages, Halt GTM and watchpoint outputs from the GTMDI module. *Figure 464* describes the TIM selection logic for generation of watchpoint messages, GTM halt and external watchpoint triggers. This logic is duplicated thus allowing independent selection of two TIM filtered inputs. *Figure 464* also shows the register bits that control the selection process. Watchpoint Logic 1 and Watchpoint Logic 2 are two implementations for the same logic.

Figure 464. TIM selection logic



Note: Figure 464 shows four instances of the TIM sub-module which corresponds to the GTM103 IP configuration. Different configurations may have different number of TIM sub-modules, thus SSEL1/2 signals have to be able to select different number of TIM sub-modules.

The selection logic for TOM and ATOM sub-modules is similar to the TIM logic. The register names are also kept the same for easy correlation of functionality.

42.6.3 GTMDI reset configuration

42.6.3.1 Enabling GTMDI class 1 operation

The GTMDI Class 1 features are always enabled after exiting the JTAG TEST-LOGIC-RESET state. But they are disabled while the JTAG is in this state. Thus, allowing low power mode. The registers are always reset while in the JTAG TEST-LOGIC-RESET state.

42.6.3.2 Enabling GTMDI class 3 operation

After exiting JTAG TEST-LOGIC-RESET state, the GTMDI classes 1 and 3 are enabled, otherwise the GTMDI class 3 features are disabled, entering in the Disable-Port Mode, thus no trace output is provided, and auxiliary port output pins are disabled.

42.6.4 Message Data Bus —interface with NAR module

42.6.4.1 Output messages

Table 526 describes the messages that the GTMDI can transmit on the Message Data Bus.

Table 526. GTMDI Nexus trace message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Debug Status Message	6	6	fixed	TCODE	Value = 0
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	32	32	fixed	STATUS	Value of the Development Status register GTMDI_DS, see *Table 527*
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message	6	6	fixed	TCODE	Value = 8
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	4	4	fixed	ETYPE	Always 0 for GTMDI
	15	15	fixed	ECODE	Error Code, see *Table 528*
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
MCS Data Trace, Data Write with Sync Message	6	6	fixed	TCODE	Value = 56
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1=MCSB)
	3	3	fixed	CHN	MCS Channel number
	14	14	fixed	ADDR	Full address of the memory location written
	32	32	fixed	DATA	Data value written
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
MCS Data Trace, Data Read with Sync Message	6	6	fixed	TCODE	Value = 57
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1=MCSB)
	3	3	fixed	CHN	MCS Channel number
	14	14	fixed	ADDR	Full address of the memory location read
	32	32	fixed	DATA	Data value read
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp

Table 526. GTMDI Nexus trace message formats(Continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Watchpoint Hit Message(1)	6	6	fixed	TCODE	Value = 15
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	15	15	fixed	WPHIT	Number indicating watchpoint source (binary coded), see *Table 529*, *Table 530* and *Table 531*
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Fetch Trace	6	6	fixed	TCODE	Value = 58
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1= MCSB)
	3	3	fixed	CHN	MCS Channel number
	6	6	fixed	FSCNT(2)	Fetch Sequential Counter. Value 0 indicates there is no relationship between the last fetch trace message and the current message. The counter rolls back to one when a non sequential address is fetched. The counter increments every time a sequential address is fetched. When counter reaches 63 a message is sent and the counter rolls back to one. If an ERROR message is sent the counter rolls back to zero.
	14	14	fixed	FADDR	First element address of the next fetch sequence
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Fetch Trace End (fetch trace was turned off, no faddr field is sent)	6	6	fixed	TCODE	Value = 59
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1= MCSB)
	3	3	fixed	CHN	MCS Channel number
	6	6	fixed	FSCNT	The message is sent with the current value in the FSCNT counter and the counter rolls back to zero after the message is sent.
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
ARU Data Trace	6	6	fixed	TCODE	Value = 60
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	CHN	ARU Debugging Channel number (0/1)
	58	58	fixed	DATA	Data value read {ARU_DATAH, ARU_DATAL}
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp

Table 526. GTMDI Nexus trace message formats(Continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
DPLL Data Trace, Data Read/Write	6	6	fixed	TCODE	Value = 61
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	24	24	fixed	DATA	Data value read/written
	12	12	fixed	ADDR	Full address of the memory location written
	1	1	fixed	RW	Read 1, Write 0
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Watchpoint Hit Message conforms to the IEEE-ISTO 5001-2003 standard, however, other messages conform to the IEEE-ISTO 5001-2011 standard.
2. FSCNT is the sequential instruction fetch counter that indicates the number of sequential instruction fetches executed by the MCS core for a channel. See *Section 42.6.9.2, Fetch trace state machine* for more information.

Table 527. Debug status field format

STATUS bit	Selected source
12:0	HS2 field of GTMDI_DS register
14:13	Reserved
15	STP bit of GTMDI_DS register
28:16	HS1 field of GTMDI_DS register
30:29	Reserved
31	HLT bit of GTMDI_DS register

Table 528. Error field format

ECODE bit	Selected source
2:0	TIM, TOM, ATOM
5:3	ARU Watchpoint, DPLL Watchpoint, SPE Watchpoint
7:6	ARU Data Trace, DPLL Data Trace
9:8	MCSA Watchpoint, MCSB Watchpoint
11:10	MCSA Data Trace, MCSB Data Trace
13:12	MCSA Fetch Trace, MCSB Fetch Trace
14	TBU Watchpoint

Table 529. Watchpoint field WPHIT[11:0] format for WPHIT[14:12]= 0b000

WPHIT bit	Selected source
0	ATOM 1 Negedge
1	ATOM 1 Posedge

Table 529. Watchpoint field WPHIT[11:0] format for WPHIT[14:12]= 0b000(Continued)

WPHIT bit	Selected source
2	ATOM 0 Negedge
3	ATOM 0 Posedge
4	TOM 1 Negedge
5	TOM 1 Posedge
6	TOM 0 Negedge
7	TOM 0 Posedge
8	TIM 1 Negedge
9	TIM 1 Posedge
10	TIM 0 Negedge
11	TIM 0 Posedge

Table 530. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b001

WPHIT bit	Selected source
0	Reserved
1	Reserved
2	Reserved
3	DPLL (memory watchpoint)
4	DPLL (SASI)
5	DPLL (TASI)
6	ARU ch1
7	ARU ch0
8	MCSB1
9	MCSB0
10	MCSA1
11	MCSA0

Table 531. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b010

WPHIT bit	Selected source
0	SPEA (DIR NEGEDGE)
1	SPEA (DIR POSEDGE)
2	SPEA (NIPD NEGEDGE)
3	SPEA (NIPD POSEDGE)
4	SPEB (DIR NEGEDGE)
5	SPEB (DIR POSEDGE)

Table 531. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b010(Continued)

WPHIT bit	Selected source
6	SPEB (NIPD NEGEDGE)
7	SPEB (NIPD POSEDGE)
11:8	Reserved

Table 532. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b011

WPHIT - bit	Selected source
0	TBU0 Watchpoint 1
1	TBU0 Watchpoint 2
2	TBU1 Watchpoint 1
3	TBU1 Watchpoint 2
4	TBU2 Watchpoint 1
5	TBU2 Watchpoint 2
11:6	Reserved

42.6.4.1.1 Temporal ordering of transmitted messages

All messages are sent out in the sequence their related event actually occurred. The GTM interface with the GTMDI provides the GTM signals to be monitored and captured. At the occurrence of any event that would cause a message, a snapshot of all information needed by the message formatter to generate a message is queued. Doing so, several events from either MCS, TIM, ATOM, ARU, etc., can be queued at the same time.

42.6.4.1.2 Interleaved messages

Two MCS cores, TIM channels and other sources can be enabled for trace. The GTMDI formats and queues all messages in the order they are generated. Since there can be message inversion among sources during the send process, if enabled by the TSFT bit in *Figure 418* or by the NAR, the messages are sent along with a timestamp. This timestamp is globally synchronized among all Nexus clients at SoC level.

42.6.5 IEEE 1149.1 (JTAG) input port

The GTMDI block uses the IEEE 1149.1 TAP for accessing registers. This port is shared among all TAP controllers on the SoC. This sharing is controlled by the value in the instruction register and client select control register. The GTMDI implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1 state machine shown in *Figure 466*. The Nexus controller state machine, shown in *Figure 467*, is defined by the IEEE-ISTO 5001-2002 standard.

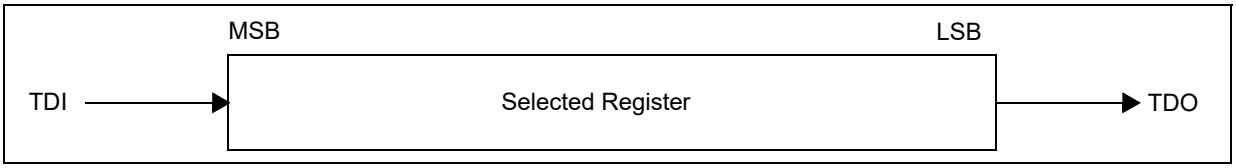
The instructions implemented by the GTMDI TAP controller are listed in *Table 533*. If any other instruction is received it is treated as a BYPASS.

Table 533. Implemented instructions

Instruction name	Private / public	Opcode	Description
NEXUS-ENABLE	Public	4b0000	Activate Nexus controller state machine to read and write GTMDI registers
BYPASS	Public	4b1111	Implements a single shift register stage providing a minimum serial length between TDI and TDO

Data is shifted between TDI and TDO starting with the least significant bit, as illustrated in *Figure 465*. This applies for the instruction register and all Nexus tool-mapped registers.

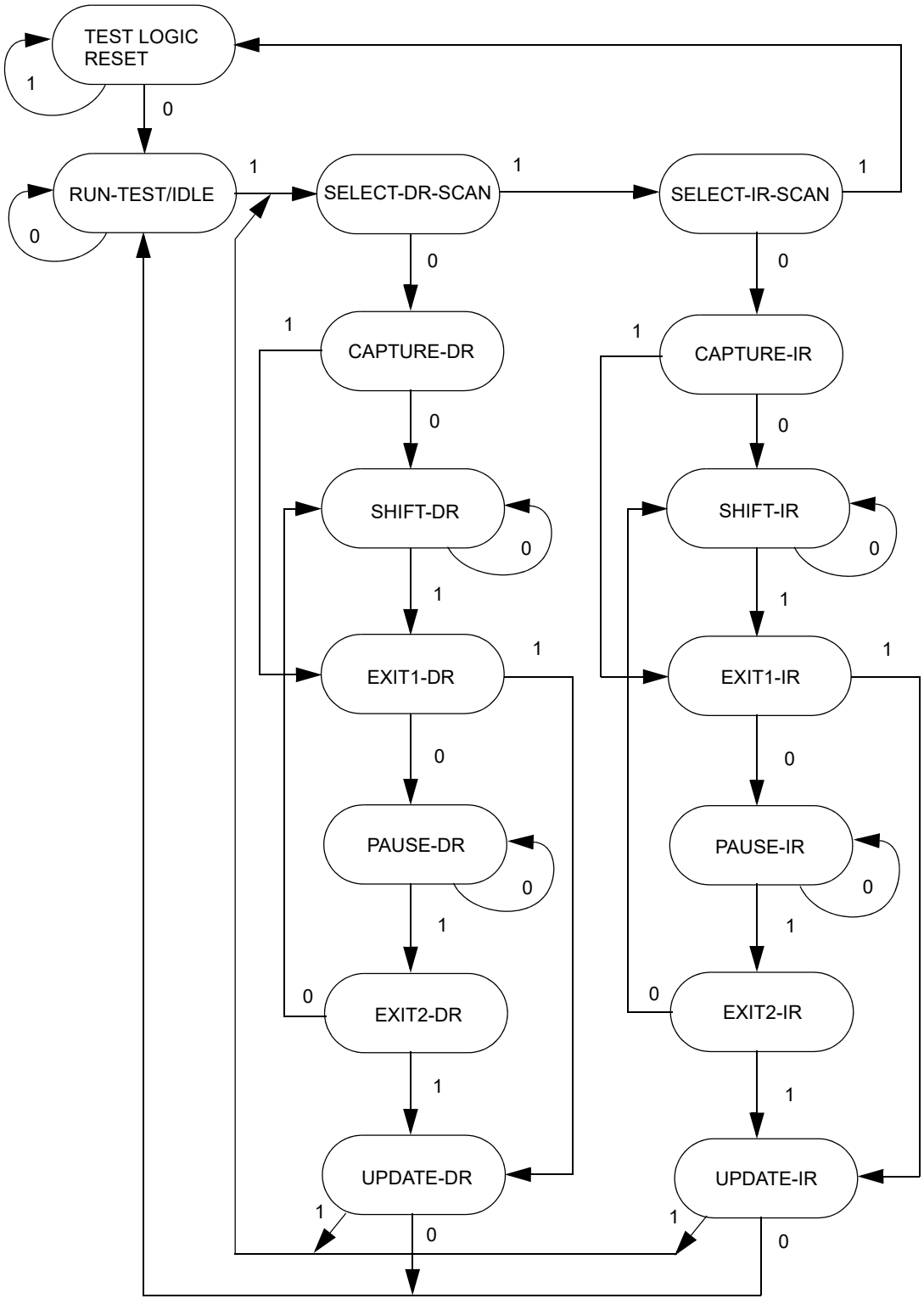
Figure 465. Shifting data into register



42.6.5.1 Enabling the GTMDI TAP controller

Assertion of the $\overline{\text{TRST}}$ input resets all TAP controllers on the SoC. The controllers are loaded with the BYPASS instruction upon exit of the TEST-LOGIC-RESET JTAG controller state. Loading the NEXUS-ENABLE instruction grants access to Nexus debug. GTMDI acts as BYPASS by loading any other instruction. The TDO bit is driven with the value of the instruction register while in SHIFT-IR state.

Figure 466. IEEE 1149.1 16-State finite state machine



NOTE: The value shown adjacent to each state transition in this figure represents the value of the TMS bit at the time of a rising edge of TCK.

42.6.5.2 Loading NEXUS-ENABLE instruction

Access to the GTMDI registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. The current instruction value is loaded into the IEEE 1149.1 shifter in the CAPTURE-IR state. The 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 467*, transitions to the REG_SELECT state. *Table 534* illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Figure 467. Nexus controller state machine

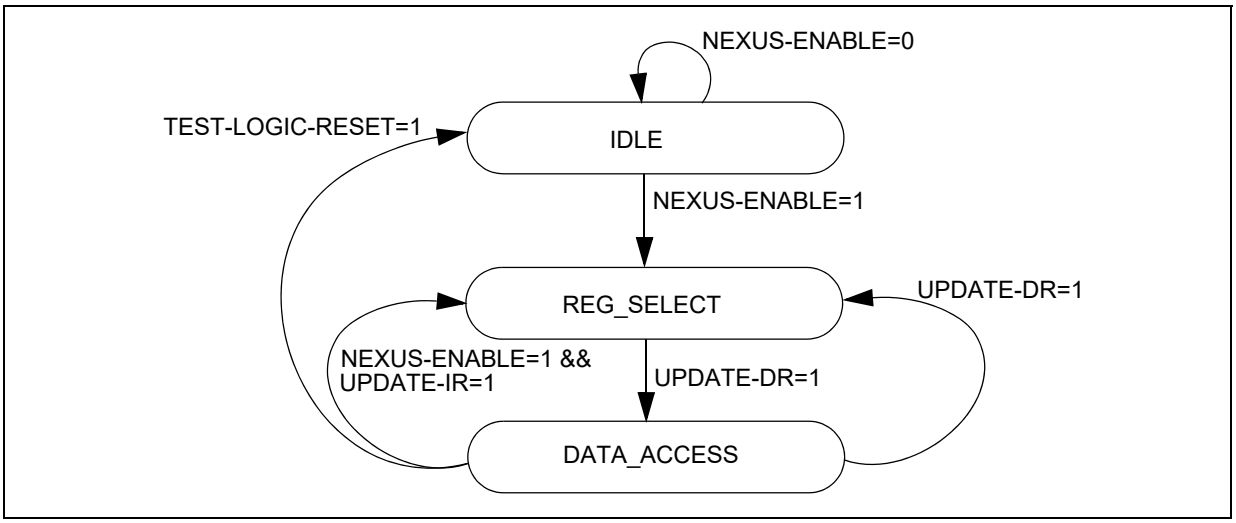


Table 534. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 State	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1 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 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction
4 TCKs				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	NEXUS-ENABLE ready to be read

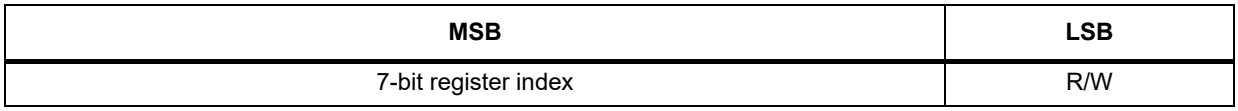
42.6.5.3 Selecting an IEEE 1149.1 register

When the NEXUS-ENABLE instruction is loaded in the GTMDI TAP controller instruction register, the input port allows development tool access to all GTMDI registers. Each register has a 7-bit address index. Some clients have registers with the same index as registers in other clients. Therefore, to access the registers of a particular client, the Client Select Control register must be set for that client.

All register access is performed via the SELECT-DR-SCAN path. The Nexus state machine defaults to the register 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 468*. The read/write control bit is set to 1 for writes and 0 for reads. The current value of the Controller Command Input is captured into the IEEE 1149.1 shifter during the CAPTURE-DR state while the Nexus controller state machine is in the REG_SELECT state.

Figure 468. 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 current register value is loaded into the IEEE 1149.1 shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

Table 535 illustrates a sequence which writes a 32-bit value to a register.

Table 535. Writing to a register

Clock	TMS	IEEE 1149.1 State	Nexus state	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1 controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in
7 TCKs				
12	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
13	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
14	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
15	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
16	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
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

42.6.6 Nexus Class 1 development support

The GTMDI contains a number of hardware hooks that aid in the development of software for the GTM MCS cores. This features described in this section are compliant with the following Nexus Class 1 features:

- Enter a debug mode at reset negation
- Enter a debug mode during normal execution
- Set halts or watchpoints

Refer to *Section 42.7, Initialization/application information* for how the following functions provide these features.

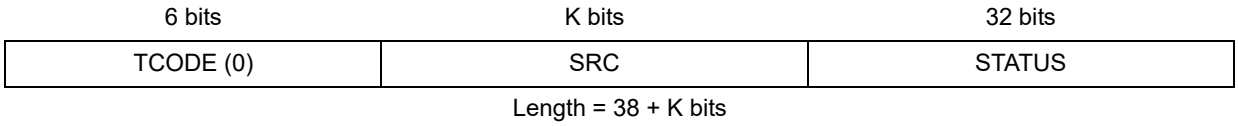
42.6.7 Debug status

The GTMDI module provides the Debug Status via the NAR SoC level modules.

42.6.7.1 Messaging

The GTMDI block provides debug status messaging using IEEE-ISTO 5001-2002 standard-defined public messages. If the message queue is full and a new message has to be sent, this message is lost and an ERROR message is queued instead, indicating the loss of trace messages. The debug status message format is shown in *Figure 469*.

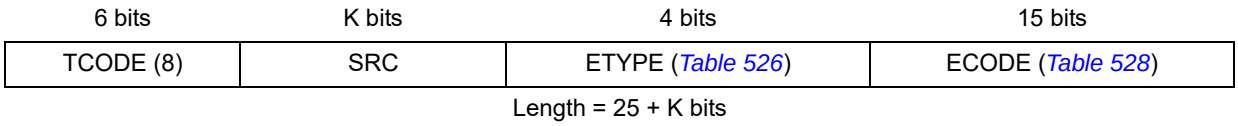
Figure 469. Debug status message format



42.6.7.2 Error messages

A debug status overrun error event is queued if a debug status event occurs while the Event Queue is full and is not able to store snapshots. The error message is stored to the event queue indicating a message was lost. The error message is sent as soon as the client is enabled for transmission. The error message format is shown in *Figure 470*.

Figure 470. Debug status error message format



Note: The timestamp value in an Error Message does not contain valid information.

42.6.8 Data trace

MCS cores, ARU and DPLL perform loads and stores to RAM memory modules. The GTMDI block gathers information about these accesses and generate trace messages if enabled.

42.6.8.1 MCS data write message

The MCS data write message contains the data write value and the address of the RAM target location. The data write message format is shown in *Figure 471*.

Figure 471. Data write message format

6 bits	K bits	1 bit	3 bits	14 bits	32 bits
TCODE (56)	SRC	MCSN	CHN	ADDR	DATA

Length = 56+K bits

42.6.8.2 MCS data read message

The MCS data read message contains the data read value and the address of the RAM target location. The data read message format is shown in *Figure 472*.

Figure 472. Data read message format

6 bits	K bits	1 bit	3 bits	14 bits	32 bits
TCODE (57)	SRC	MCSN	CHN	ADDR	DATA

Length = 56+K bits

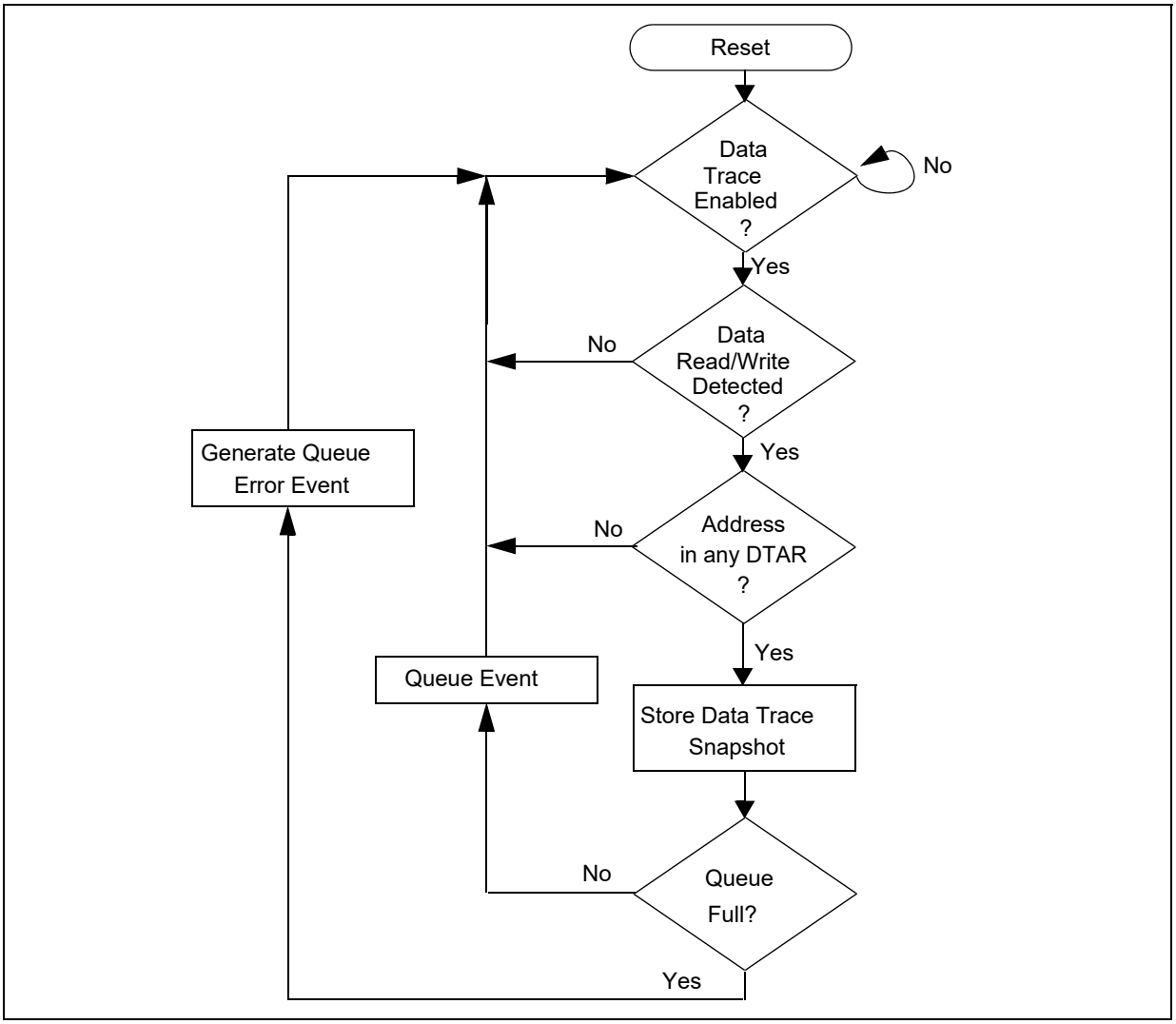
42.6.8.3 MCS data trace operation

Data tracing is performed by snooping a dedicated MCS/Nexus interface for RAM read and write cycles. Data trace functions are enabled by setting the appropriate fields in the following registers:

- MCSx Development Control (GTMDI_MCSx_DC) Register
- MCSx Data Trace Control (GTMDI_MCSx_DTC) Register
- MCSx Data Trace Address Range (GTMDI_MCSx_DTAR) Register

For details on register configuration, refer to *Section 42.5.1, Register descriptions*. The MCSx data tracing have independent Data Trace Address Range registers which allows more flexibility. An Error condition may occur in case of Queue full event. Data trace flow is depicted in *Figure 473*.

Figure 473. GTMDI generic data trace flow diagram



42.6.8.3.1 MCS data trace windowing

Data trace windowing is provided so that the development tool can decrease the auxiliary port usage by limiting the accesses that are traced. Data trace windowing is achieved via the address range defined by the DTSA and the DTEA fields of the DTAR register. RAM accesses are traced if their address fall in any of these ranges and the specific range is enabled in the data trace control register. Data read and/or data write trace may be enabled via the Read/Write Trace Control fields in the data trace control registers.

Data trace ranges are 32-bit aligned. This alignment is done by making the two least significant bits of the DTSA and DTEA fields read only. Since the two least significant bits of DTSA and DTEA are read only and default to different values, DTSA can never be equal to DTEA. *Figure 474* shows the relationship between the DTSA and DTEA fields.

Figure 474. Data trace address range options

Programmed Value	Range Selected
DTSA =< DTEA	[DTSA: DTEA]
DTSA > DTEA	All addresses are out of range(1)

1. Since all addresses are considered to be out of range, a Data Trace Event may be recognized if the DTC register is programmed so that addresses out of range are queued.

Note: Before sending through the NAR interface, the MCS, DPLL or ARU Data Trace message is buffered. Due to bandwidth constraints in the NAR interface and in the data trace buffering, this data trace message can be lost when superseded by another data trace message from the same source. In this case an error message is sent. See Table 526 for more details about GTMDI NAR messages.

42.6.8.4 DPLL data trace

DPLL Data Trace is a straight forward implementation in which the trace is enabled using the DMC bit in the *Figure 435*. It is possible to select individual DPLL RAM modules such as RAM1a, RAM1b or RAM2 for tracing using the RAMSEL bit in the same register. Only one memory module is selected for tracing at a given time. Start/Stop inputs from the SPU module can be used to control the DPLL Data Trace messages as well as JTAG interface. This is achieved by controlling the DMC bit which can be done from SPU or JTAG interface. See *Table 498* for more details about the bits that should be used in this process.

42.6.8.5 ARU data trace

The GTM module selects two ARU addresses to be monitored, using the ARU Debugging Channels 0 and 1, see the GTM documentation for details on how to program these channels. The data trace for these two channels is controlled by DMC1/2 bits in the GTMDI_ARU_DTC, see *Section 42.5.1.26, ARU data trace control register (GTMDI_ARU_DTC)*. Once enabled, messages are generated as soon as a valid data is identified in one of the ARU Debugging Channels, 0 or 1.

42.6.9 Fetch trace

Fetch trace tries to record all instruction fetch accesses done by an MCS channel. Fetch trace is handled independently for each MCS channel, therefore what happens to a certain channel does not affect other channels since all channels run independently. Even error conditions are handled independently for each channel.

Fetch trace can be enabled/disabled by two ways:

- By watchpoint hit, if the watchpoint is configured to do so
- Directly by register

When a fetch access hits a watchpoint and the last enables fetch trace (having it disabled before the watchpoint hit), that access is ignored by the fetch trace logic.

42.6.9.1 Enabling MCS program fetch trace

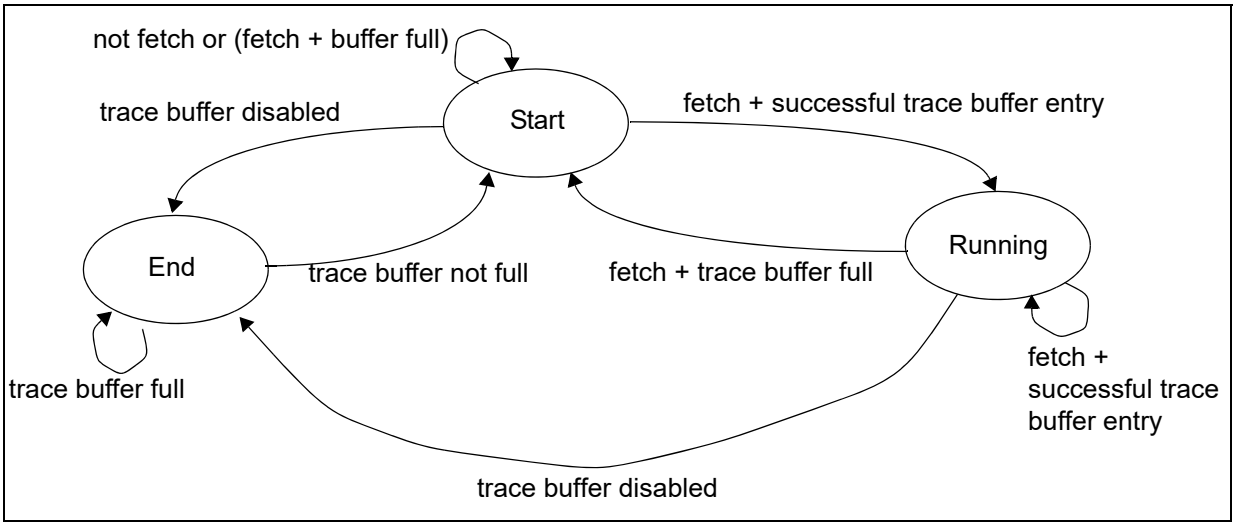
Program fetch trace for the MCS is enabled on a channel by channel basis. This selective enabling allows the debug tool to control the bandwidth for the transmitted messages. The program fetch trace is enabled through a JTAG register, see *Figure 445*.Since the registers

may be changed in the middle of a program execution, program fetch trace may be enabled or disabled during program execution as well.

42.6.9.2 Fetch trace state machine

The fetch trace state machine is composed basically by three states: *start*, *running* and *end*. When the fetch trace is enabled, fetch state machine goes to the *start* state. *Figure 475* describes the fetch trace machine state diagram.

Figure 475. Fetch trace state machine state diagram



42.6.9.2.1 Fetch trace start state

In this state, the fetch trace logic waits for a fetch access to occur. When a fetch access occurs, the fetch trace logic tries to insert (into the corresponding MCS trace buffer) a fetch trace message with FSCNT=0 and the FADDR field equal to the fetch access address. It is possible that the corresponding MCS trace message buffer is full when fetch trace logic tries to insert the message. In this case, the fetch trace logic ignores the current fetch access, does not generate any error, and tries again when another fetch access occurs.

After inserting a message into the buffer, the fetch trace state machine goes to the *running* state.

It is also possible that the fetch trace is disabled before a message is inserted into the MCS trace buffer. In this case state machine goes to the *end* state

42.6.9.2.2 Fetch trace running state

When the fetch trace state machine enters in the *running state* a counter, called CNT, is initialized to 1 and starts counting the number of sequential instruction fetch accesses.

When a fetch access with address ADDR is detected, the fetch trace logic tests if ADDR is equal to the previous fetch address, ADDR_PRV, +1 (ADDR = ADDR_PRV + 1). If so, no message is buffered for transmission and CNT increments by 1 (CNT = CNT + 1). Otherwise, if ADDR is not a sequential address, a fetch trace message is generated with the FSCNT field equals to the CNT counter value (FSCNT = CNT) and FADDR equals to the current address ADDR. The CNT counter is then initialized to 1.

At the moment the previous message is generated, the corresponding MCS fetch trace buffer may or may not be full. If the buffer is not full, the message is inserted into the buffer and the fetch trace state machine continues in the *running* state, otherwise a fetch trace error message is generated for the corresponding MCS and the fetch trace state machine goes to the *start* state.

Note: The MCS is a multi-thread core which executes eight threads, also called channels, in fixed time-slots of one system clock for each thread. Fetch trace is a per channel resource. Thus if more than one channel is selected for trace, each one generates its corresponding fetch trace message. There is one independent fetch state machine and CNT counter for each selected channel.

42.6.9.2.3 Fetch trace end state

In this state, the fetch trace logic ignores all fetch accesses and waits for the corresponding MCS trace buffer to be not full. When this happens, a fetch trace end message is inserted into the buffer with FSCNT equal to CNT (FSCNT = CNT). The CNT value is always reset to 0 when previous state was *start* state.

If the buffer is always full, the fetch trace logic does not exit *end* state until a reset occurs.

Note: A Fetch trace end message is sent when a system reset is detected in the GTM module. There are no further indications about the GTM reset condition.

42.6.10 Watchpoint trace

The GTMDI module provides watchpoint messaging via the auxiliary port, as defined by the IEEE-ISTO 5001-2002 standard.

42.6.10.1 Messaging

The GTMDI block provides watchpoint messaging using IEEE-ISTO 5001-2002 standard-defined public messages. When a watchpoint occurs, a watchpoint event is sent to the Event Queue. If the watchpoint condition occurs while the Event Queue is full, an error message is generated. The watchpoint message format is shown in *Figure 476*.

Figure 476. Watchpoint hit message format

6 bits	K bits	15 bits
TCODE (15)	SRC	WPHIT

Length = 21+Kbits

The values for the WPHIT packet are described in *Table 529*, *Table 530* and *Table 531*. Notice that WPHIT is never 0 because the message is only generated when a watchpoint has occurred.

42.7 Initialization/application information

42.7.1 Accessing GTMDI tool-mapped registers

To initialize the JTAG port for register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Retrieve Device ID if needed
3. Load the Nexus TAP controller with the NEXUS-ENABLE instruction

To write control data to GTMDI tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select register with a pass through the SELECT-DR-SCAN path in the JTAG state machine
2. Write the register value with a pass through the SELECT-DR-SCAN path. The prior value of this register is shifted out during the write

To read status and control data from GTMDI tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear write bit to select register with a pass through SELECT-DR-SCAN path
2. Read the register value with a pass through the SELECT-DR-SCAN path. Data shifted in is ignored

See *Section 42.6.5, IEEE 1149.1 (JTAG) input port* for more detail.

43 GTM101 Integration (GTMINT) Module

43.1 Introduction

This chapter specifies the functionality of the GTMINT module. The GTMINT is integrated IP with timer and debug functions and is composed of:

- The Bosch GTM-IP_101 timer module (also called GTM-IP)
- The GTMDI debug interface IP
- 8 ECC error correction encoder / decoders for the RAMs required by the timer.

The timer GTM-IP_101 is a very complex block that can be configured to perform several timer functions. The debugger module is the GTMDI IP (GTM Development Interface IP) that is a Nexus based interface which is connected to the chip level NAR and used to monitor the software and hardware activities of GTM-IP. The ECC sub-blocks generate the parity bits on RAM writes and detect errors in the data read from the RAMs.

Note: This chapter does not intend to present details of GTM-IP nor GTMDI. For full information about the timer and the debugger, refer to its respective documentation.

43.1.1 Features

The GTMINT module features[I] include:

- GTM-IP timer features
  - Time base generators
  - Advanced data signal / time base router
  - Input signal processors (16 inputs)
  - Simple output signal generators (32 outputs)
  - Microprocessed output signal generators (24 outputs)
  - Digital PLL
  - BLDC support
  - Safety features – GTM is a non-safety relevant module
    - Output compare unit
    - Monitoring unit
  - CPU bus interface clock operates asynchronously to timer section clock allowing CPU low power operation.
- GTMDI debug block features
  - Full duplex pin interface for medium and high visibility throughput
    - Pin interface shared among all Nexus-based development interfaces on the device
    - Auxiliary Output Port
      DATA with MDO (Message Data Out) and MSEO (Message Start/End)
      2 $\overline{\text{EVTO}}$ (Event Out) pins for GTM real time signal monitoring
    - Auxiliary Input Port
      1 $\overline{\text{EVTI}}$ (Event In) pin

I. Configuration of GTM has to be confirmed for final silicon

  - IEEE 1149.1 (JTAG) Test Access Port (TAP)
    Support for optional Multi-JTAG TAP Linking Module (TLM)
    4 pins (TDI, TDO, TMS, and TCK)
    Reset input $\overline{\text{TRST}}$ driven by either the Nexus Port Controller or an external pin
- GTM Development Support
  - Read/Write registers in debug mode from JTAG port
  - Ability to enter debug mode at reset negation or during normal execution
  - Breakpoint and Watchpoint Configuration
  - Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This allows the development tool to trace reads and writes to selected RAM address ranges
    Two data trace windows for each MCS core with programmable address ranges and access attributes are provided. Data trace windowing reduces the requirement on the Aurora port bandwidth by constraining the number of trace locations.
    Fetch trace capability controlled by two programmable Watchpoints.
    Private messages are used to indicate special MCS core cases not covered by public messages. Up to two[(m)] MCS cores can be debugged at the same time.
  - Watchpoint Messaging (WPM)
- Trigger Control
  - The GTMDI provides selection of trigger signals that will generate debug actions at system level such as Halt GTM itself or cores in the same device.
  - Trace enable signals are provided to control the start/stop of trace messages for each one of the GTM Nexus message sources, such as TIM, TOM, ATOM, etc.
- GTMDI Clock operation
  - The module is able to operate in any frequency ratio considering GTMDI and JTAG frequency. The GTMDI should operate in the same clock source as the GTM IP module.

- ECC encoder / decoder for each of 8 RAM interfaces
  - Parity calculation using the whole word and the target memory address value.
  - Detection and correction of single error on the read data value
  - Detection of single bit error on address field but no need for correction
  - Double error detection on data or address field without error correction.
  - ECC report generation for each RAM containing:
    - Address of the error in the register map (see *Section 43.2, Memory map and register definition*)
    - Non-correctable error indication (not correctable data + address ECC detected errors)

m. Note that by enabling two MCS cores at the same time for data trace can exceed the capacity of the NAR Client interface throughput thus information can be lost.

  - Correctable error indication (correctable data error)
- Generation of 103 interrupt requests and or DMA requests.
- Debugger is allowed to read all GTM registers through the peripheral bus even when the timer is running. Read-access of the debugger is possible without destroying or changing the contents of these registers.
- Debugger is allowed to read all GTM addresses read-/writeable from the peripheral bus when GTM is running or halted.

43.1.2 Modes of operation

43.1.2.1 Normal Mode

The normal mode is the mode in which the GTMINT module is after a reset is applied. In this mode the timer IP is functional and processes input timing signals and generates timed outputs based on the input timing signals or controlled by the internal channels software. The GTMDI development interface is also active.

43.1.2.2 Stop Mode

This mode is set when the global stop input is received or when field GTMMCR[MDIS] is asserted. In this case all functional clocks are disabled (ipg_clk and ipg_clk_gtm), but the slave bus clock is not disabled. See more details on *Section 43.3.4, Stop Mode Description*.

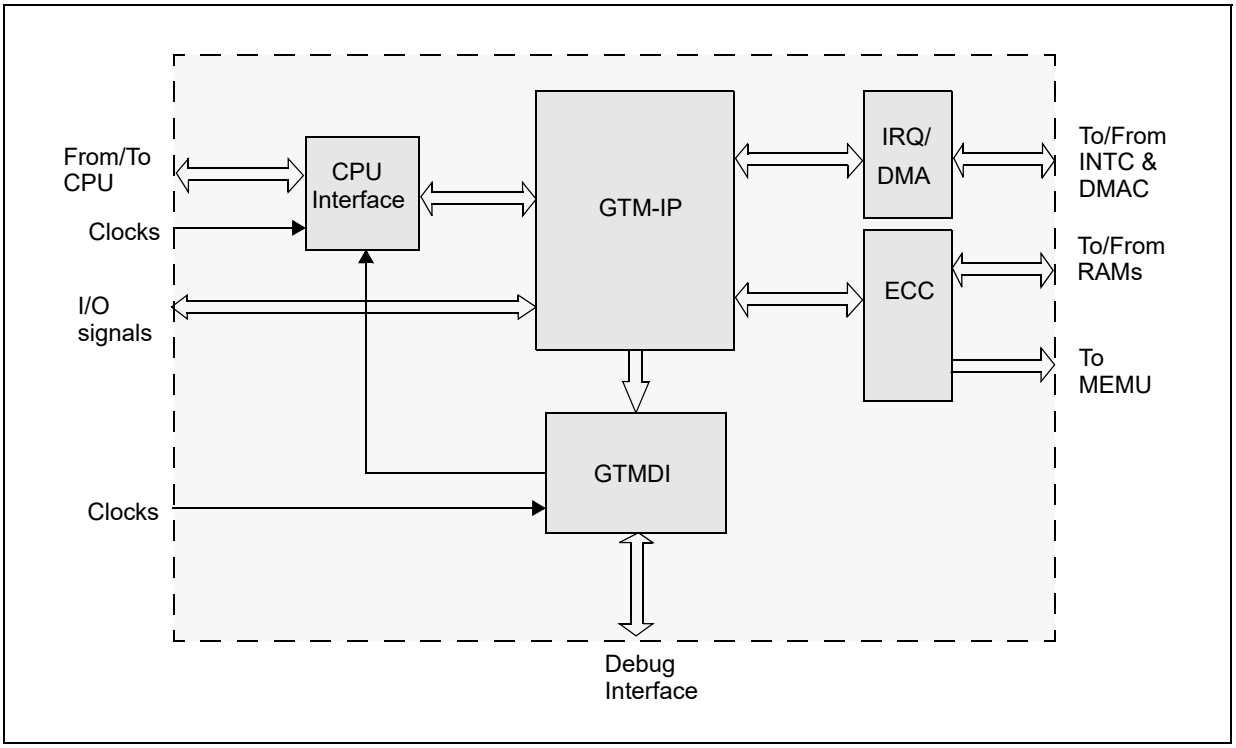
43.1.2.3 Debug Mode (or Halt Mode)

The debug mode or GTM-IP halt mode is externally requested through the GTMDI. In this mode, the GTM-IP main clock is requested to stop (ipg_clk_gtm only) and the bus interface clocks remain running to access all registers and memories. It is possible to read the registers not interfering into the logic status, even for FIFOs, by configuring the debug access controls on GTMDI.

43.1.3 Block diagram

Figure 477 shows the interconnections in the GTMINT module.

Figure 477. GTMINT block diagram



43.2 Memory map and register definition

This section provides a reference table to the GTM-IP submodule registers. Details of each register are found in the *GTM-IP specification* document. Additional registers from those used in the GTM-IP are described in this section.

43.2.1 Module memory map

The GTMINT simplified memory map is shown in *Table 536*. Offsets are relative to the GTMINT base address. See the device memory map for the module base address.

Note: The GTMINT module has only a few registers, detailed in Section 43.2.2, Register descriptions. The module also acts as an interface to all implemented GTM-101 control registers. For details see the GTM-IP Specification document.

Table 536. High level memory map

Offset	Use(1)
0x00000000	GTM-IP TOP-Level Configuration Registers
0x000000C0	GTM-IP Integration Registers
0x00000100	TBU
0x00000180	MON
0x00000200	CMP
0x00000280	ARU

Table 536. High level memory map(Continued)

Offset	Use[1]
0x00000300	CMU
0x00000600	ICM
0x00000800	SPE0
0x00000880	SPE1
0x00000F00	MAP
0x00000F40	MCFG
0x00001000	TIM0
0x00001800	TIM1
0x00008000	TOM0
0x00008800	TOM1
0x0000D000	ATOM0
0x0000D800	ATOM1
0x0000E000	ATOM2
0x00018000	F2A0
0x00018080	AFD0
0x00018400	FIFO0
0x00019000	FIFO0_MEMORY
0x00028000	DPLL
0x00028200	DPLL_MEM R1A
0x00028400	DPLL_MEM R1B
0x00028600	DPLL_MEM_REGION R1C
0x00028E00	DPLL REGION EXT
0x0002C000	DPLL_MEM R2
0x00030000	MCS0
0x00031000	MCS1
0x00038000	MCS0_MEMORY
0x00040000	MCS1_MEMORY

1. Configuration of GTM has to be confirmed for final silicon

This section presents the GTMINT Module Memory Map (that is basically the same of the timer GTM-IP) in *Table 537*.

Table 537. GTMINT module memory map

Offset	Use	Access	Reset value[1]	Location
GTM-IP TOP-Level Configuration Registers				
0x00000	GTM-IP Version control register (GTM_REV)	R	0x101X_XXXX (2)	See GTM Spec.(2)
0x00004	GTM-IP Global reset register (GTM_RST)	R/W	0x0000_0000	See GTM Spec.(2)
0x00008	GTM-IP Global control register (GTM_CTRL)	R/W	0x0000_0001	See GTM Spec.(2)
0x0000C	GTM-IP Timeout Address Register (GTM_AEI_ADDR_XPT)	R	0x0000_0000	See GTM Spec.(2)
0x00010	GTM-IP Interrupt Notification Register (GTM_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x00014	GTM-IP Interrupt Enable (GTM_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00018	GTM-IP Interrupt Force (GTM_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x0001C	GTM-IP Interrupt Mode (GTM_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x00020	GTM-IP (GTM_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00024 – 0x0002C	Reserved			
0x00030	GTM-IP AEI Bridge Mode (GTM_BRIDGE_MODE)	R/W	0x0400_0001 (3)	See GTM Spec.(2)
0x00034	GTM-IP AEI Bridge Pointer1 (GTM_BRIDGE_PTR1)	R	0x0040_0000 (3)	See GTM Spec.(2)
0x00038	GTM-IP AEI Bridge Pointer2 (GTM_BRIDGE_PTR2)	R	0x0000_0000	See GTM Spec.(2)
0x0003C	Reserved			
0x00040+m*0x04	GTM-IP TIM[m] auxiliary source selection register (m = 0..1) (GTM_TIM[m]_AUX_IN_SRC)	R/W	0x0000_0000	See GTM Spec.(2)
0x00048 – 0x00058	Reserved			
0x0005C – 0x000BC	Reserved			
Top GTM Integration Registers				
0x000C0	GTMMCR – GTM Module Configuration Register	R/W	0xX000_X000 (3)	*on page 947*
0x000C4	Reserved			
0x000C8	GTMINTCLR – GTM Interrupts Clear Register	R/W	0x0000_0000	*on page 948*
0x000CC	GTMMAEICR – GTM AEI Control Register	R/W	0x0000_0000	*on page 950*
0x000D0 – 0x000FC	Reserved			

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
Time Base Unit (TBU) Configuration Registers				
0x00100	TBU global channel enable (TBU_CHEN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00104+m*0x08	TBU channel m control (m = 0..2) (TBU_CH[m]_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x00108+n*0x08	TBU channel n base (n = 0...2) (TBU_CH[n]_BASE)	R/W	0x0000_0000	See GTM Spec.(2)
0x0011C – 0x00120	Reserved for TBU			
0x00124 – 0x0017C	Reserved			
Monitor Unit (MON) Configuration Registers				
0x00180	Monitor Status register (MON_STATUS)	R/W	0x0000_0000	See GTM Spec.(2)
0x00184	Monitor activity register 0 (MON_ACTIVITY_0)	R/W	0x0000_0000	See GTM Spec.(2)
0x00188 – 0x001A4	Reserved for MON			
0x001A8 – 0x001FC	Reserved			
Output Compare Unit (CMP) Configuration Registers				
0x00200	Comparator enable register (CMP_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00204	Event notification register (CMP_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x00208	Interrupt enable register (CMP_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x0020C	Interrupt force register (CMP_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x00210	IRQ mode configuration register (CMP_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x00214	IRQ mode configuration register (CMP_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00218 – 0x00220	Reserved for CMP			
0x00224 – 0x0027C	Reserved			
Advanced Routing Unit (ARU) Configuration Registers				
0x00280	ARU access register (ARU_ACCESS)	R/W	0x0000_01FE	See GTM Spec.(2)
0x00284	ARU access register upper data word (ARU_DATA_H)	R/W	0x0000_0000	See GTM Spec.(2)
0x00288	ARU access register lower data word (ARU_DATA_L)	R/W	0x0000_0000	See GTM Spec.(2)

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x0028C	Debug access channel 0 (ARU_DBG_ACCESS0)	R/W	0x0000_01FE	See GTM Spec.(2)
0x00290	Debug access 0 transfer register upper data word (ARU_DBG_DATA0_H)	R	0x0000_0000	See GTM Spec.(2)
0x00294	Debug access 0 transfer register lower data word (ARU_DBG_DATA0_L)	R	0x0000_0000	See GTM Spec.(2)
0x00298	Debug access channel 1 (ARU_DBG_ACCESS1)	R/W	0x0000_01FE	See GTM Spec.(2)
0x0029C	Debug access 1 transfer register upper data word (ARU_DBG_DATA1_H)	R	0x0000_0000	See GTM Spec.(2)
0x002A0	Debug access 1 transfer register lower data word (ARU_DBG_DATA1_L)	R	0x0000_0000	See GTM Spec.(2)
0x002A4	ARU Interrupt notification register (ARU_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x002A8	ARU Interrupt enable register (ARU_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x002AC	Register for forcing the ARU_NEW_DATA_IRQ interrupt (ARU_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x002B0	IRQ mode configuration register (ARU_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x002B4 – 0x002FC	Reserved			
Clock Management Unit (CMU) Configuration Registers				
0x00300	Clock enable (CMU_CLK_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00304	Global clock control numerator (CMU_GCLK_NUM)	R/W	0x0000_0001	See GTM Spec.(2)
0x00308	Global clock control denominator (CMU_GCLK_DEN)	R/W	0x0000_0001	See GTM Spec.(2)
0x0030C+n*0x04	Control for clock source n (n = 0...7) (CMU_CLK_[n]_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0032C+m*0x08	External clock m control numerator (m = 0...2) (CMU_ECLK_[m]_NUM)	R/W	0x0000_0001	See GTM Spec.(2)
0x00330+m*0x08	External clock m control denominator (m = 0...2) (CMU_ECLK_[m]_DEN)	R/W	0x0000_0001	See GTM Spec.(2)
0x00344	Global clock control denominator (CMU_FXCLK_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x00348 – 0x003FC	Reserved			
0x00400 – 0x00474	Reserved for BRC			

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x00478 – 0x005FC	Reserved			
Interrupt Concentrator Module (ICM) Configuration Registers				
0x00600	ICM Interrupt group register covering infrastructural and safety components (ARU, BRC, AEI, PSM, MAP, CMP, SPE) (ICM_IRQG_0)	R	0x0000_0000	See GTM Spec.(2)
0x00604	ICM Interrupt group register covering DPLL (ICM_IRQG_1)	R	0x0000_0000	See GTM Spec.(2)
0x00608	ICM Interrupt group register covering TIM0, TIM1, TIM2 (ICM_IRQG_2)	R	0x0000_0000	See GTM Spec.(2)
0x0060C	Reserved			
0x00610	ICM Interrupt group register covering MCS0 to MCS2 submodules (ICM_IRQG_4)	R	0x0000_0000	See GTM Spec.(2)
0x00614	Reserved			
0x00618	ICM Interrupt group register covering GTM-IP output submodules TOM0 to TOM1 (ICM_IRQG_6)	R	0x0000_0000	See GTM Spec.(2)
0x0061C – 0x00620	Reserved for ICM			
0x00624	ICM Interrupt group register covering GTM-IP output submodules ATOM0, ATOM1, ATOM2 and ATOM3 (ICM_IRQG_9)	R	0x0000_0000	See GTM Spec.(2)
0x00628 – 0x0062C	Reserved for ICM			
0x00630	ICM Interrupt group register for module error interrupt information (ICM_IRQG_MEI)	R	0x0000_0000	See GTM Spec.(2)
0x00634	ICM Interrupt group register 0 for FIFO channel error interrupt information (ICM_IRQG_CEI0)	R	0x0000_0000	See GTM Spec.(2)
0x00638	ICM Interrupt group register 1 for TIM0..2 channel error interrupt information (ICM_IRQG_CEI1)	R	0x0000_0000	See GTM Spec.(2)
0x0063C	Reserved for ICM			
0x00640	ICM Interrupt group register 3 for MCS0..2 channel error interrupt information (ICM_IRQG_CEI3)	R	0x0000_0000	See GTM Spec.(2)
0x00644	Reserved for ICM			
0x00648 – 0x007FC	Reserved			
Sensor Pattern Evaluation (SPE) Registers (AAi = i*0x80**, i = 0..1)**				
0x00800 + AAi	SPE Control status register (SPE[i]_CTRL_STAT)	R/W	0x0000_0000	See GTM Spec.(2)

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x00804 + AAi	SPE Input pattern definition register (SPE[i]_PAT)	R/W	0x0000_0000	See GTM Spec.(2)
0x00808+n*0x04 + AAi	SPE Output definition registers (SPE[i]_OUT_PAT[n]), n = 0...7	R/W	0x0000_0000	See GTM Spec.(2)
0x00828 + AAi	SPE output control register (SPE[i]_OUT_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0082C + AAi	SPE Interrupt notification register (SPE[i]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x00830 + AAi	SPE Interrupt enable register (SPE[i]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00834 + AAi	SPE Interrupt generation by software (SPE[i]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x00838 + AAi	IRQ mode configuration register (SPE[i]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x0083C + AAi	SPE Interrupt enable register (SPE[i]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00840 + AAi	SPE Interrupt generation by software (SPE[i]_REV_CNT)	R/W	0x0000_0000	See GTM Spec.(2)
0x00844 + AAi	IRQ mode configuration register (SPE[i]_REV_CMP)	R/W	0x0000_0000	See GTM Spec.(2)
(0x00848 – 0x0087C) + AAi	Reserved			
0x00900 – 0x009FC	Reserved for SPE2/3			
0x00A00 – 0x00EFC	Reserved			
TIM0 Input Mapping Module (MAP) Registers				
0x00F00	MAP Control register (MAP_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x00F04 – 0x00F3C	Reserved for MAP			
Memory Configuration (MCFG) Configuration Registers				
0x00F40	Memory layout configuration (MCFG_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x00F44 – 0x00FA0	Reserved for MCFG			
0x00FA4 – 0x00FFC	Reserved			
Timer Input Module (TIM) Configuration Registers (BBi = i*0x800, n = 0...7, i = 0..1)				
0x01000+n*0x80 + BBi	Channel n general purpose 0 (TIM[i]_CH[n]_GPR0)	R	0x0X00_0000	See GTM Spec.(2)
0x01004+n*0x80 + BBi	Channel n general purpose 1 (TIM[i]_CH[n]_GPR1)	R	0x0X00_0000	See GTM Spec.(2)

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x01008+n*0x80 + BBi	Channel n SMU counter (TIM[i]_CH[n]_CNT)	R	0x0000_0000	See GTM Spec.(2)
0x0100C+n*0x80 + BBi	Channel n SMU counter (TIM[i]_CH[n]_ECNT)	R	0x0000_0000	See GTM Spec.(2)
0x01010+n*0x80 + BBi	Channel n SMU shadow counter (TIM[i]_CH[n]_CNTS)	R/W	0x0X00_0000	See GTM Spec.(2)
0x01014+n*0x80 + BBi	Channel n TDU control (TIM[i]_CH[n]_TDUC)	R/W	0x0000_0000	See GTM Spec.(2)
0x01018+n*0x80 + BBi	Channel n TDU control (TIM[i]_CH[n]_TDUV)	R/W	0x0000_0000	See GTM Spec.(2)
0x0101C+n*0x80 + BBi	Channel n filter parameter 0 (TIM[i]_CH[n]_FLT_RE)	R/W	0x0000_0000	See GTM Spec.(2)
0x01020+n*0x80 + BBi	Channel n filter parameter 1 (TIM[i]_CH[n]_FLT_FE)	R/W	0x0000_0000	See GTM Spec.(2)
0x01024+n*0x80 + BBi	Channel n control (TIM[i]_CH[n]_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x01028+n*0x80 + BBi	Channel n control (TIM[i]_CH[n]_ECTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0102C+n*0x80 + BBi	Channel n interrupt notification (TIM[i]_CH[n]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x01030+n*0x80 + BBi	Channel n interrupt enable (TIM[i]_CH[n]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x01034+n*0x80 + BBi	Channel n software interrupt force (TIM[i]_CH[n]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x01038+n*0x80 + BBi	Channel n IRQ mode configuration register (TIM[i]_CH[n]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x0103C+n*0x80 + BBi	Channel n software interrupt force (TIM[i]_CH[n]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
(0x01040+n*0x80 – 0x01074+n*0x80) + BBi	Reserved for TIM[i]_CH[n]			
0x01078 + BBi	Channel n IRQ mode configuration register (TIM[i]_IN_SRC)	R/W	0x0000_0000	See GTM Spec.(2)
0x107C + BBi	TIM Global software reset (TIM[i]_RST)	R/W	0x0000_0000	See GTM Spec.(2)
(0x01078+n*0x80 – 0x0107C+n*0x80) + BBi	Reserved (n > 0)			
(0x01400 – 0x017FC) + BBi	Reserved			
0x02000 – 0x04400	Reserved for TIM2 to TIM6			
0x04404 – 0x07FFC	Reserved			

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
Timer Output Module (TOM) Configuration Registers (CCi = i*0x800, **i = 0...1, n = 0...15)**				
0x08000 + n*0x0040 + CCi	TOM Channel m control register (TOM[i]_CH[n]_CTRL), i = 0...1, n = 0...15	R/W	0x0000_X000	See GTM Spec.(2)
0x08004 + n*0x0040 + CCi	TOM Channel n CCU0 compare shadow register (TOM[i]_CH[n]_SR0)	R/W	0x0000_0000	See GTM Spec.(2)
0x08008 + n*0x0040 + CCi	TOM Channel n CCU1 compare shadow register (TOM[i]_CH[n]_SR1)	R/W	0x0000_0000	See GTM Spec.(2)
0x0800C + n*0x0040 + CCi	TOM Channel n CCU0 compare register (TOM[i]_CH[n]_CM0)	R/W	0x0000_0000	See GTM Spec.(2)
0x08010 + n*0x0040 + CCi	TOM Channel n CCU1 compare register (TOM[i]_CH[n]_CM1)	R/W	0x0000_0000	See GTM Spec.(2)
0x08014 + n*0x0040 + CCi	TOM Channel n CCU0 counter register (TOM[i]_CH[n]_CN0)	R/W	0x0000_0000	See GTM Spec.(2)
0x08018 + n*0x0040 + CCi	TOM channel n status (TOM[i]_CH[n]_STAT)	R	0x0000_0001	See GTM Spec.(2)
0x0801C + n*0x0040 + CCi	TOM channel n interrupt notification register (TOM[i]_CH[n]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x08020 + n*0x0040 + CCi	TOM channel n interrupt enable register (TOM[i]_CH[n]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x08024 + n*0x0040 + CCi	TOM channel n software interrupt generation (TOM[i]_CH[n]_IRQ_FORCINT)		0x0000_0000	See GTM Spec.(2)
0x08028 + n*0x0040 + CCi	Channel n IRQ mode configuration register (TOM[i]_CH[n]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x0802C + n*0x0040 + CCi	Reserved			
0x08030 + CCi	TGC0 global control reg (TOM[i]_TGC0_GLB_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x08034 + CCi	TGC0 action time base register (TOM[i]_TGC0_ACT_TB)	R/W	0x0000_0000	See GTM Spec.(2)
0x08038 + CCi	TGC0 force update control reg (TOM[i]_TGC0_FUPD_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0803C + CCi	TGC0 internal trigger control (TOM[i]_TGC0_INT_TRIG)	R/W	0x0000_0000	See GTM Spec.(2)
0x08070 + CCi	TGC0 enable/disable control reg (TOM[i]_TGC0_ENDIS_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x08074 + CCi	TGC0 enable/disable status reg (TOM[i]_TGC0_ENDIS_STAT)	R/W	0x0000_0000	See GTM Spec.(2)
0x08078 + CCi	TGC0 output enable control reg (TOM[i]_TGC0_OUTEN_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0807C + CCi	TGC0 output enable status reg (TOM[i]_TGC0_OUTEN_STAT)	R/W	0x0000_0000	See GTM Spec.(2)

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x08230 + CCi	TGC1 global control register (TOM[i]_TGC1_GLB_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x08234 + CCi	TGC1 action time base register (TOM[i]_TGC1_ACT_TB)	R/W	0x0000_0000	See GTM Spec.(2)
0x08238 + CCi	TGC1 force update control reg (TOM[i]_TGC1_FUPD_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0823C + CCi	TGC1 internal trigger control (TOM[i]_TGC1_INT_TRIG)	R/W	0x0000_0000	See GTM Spec.(2)
0x08270 + CCi	TGC1 enable/disable control reg (TOM[i]_TGC1_ENDIS_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x08274 + CCi	TGC1 enable/disable status reg (TOM[i]_TGC1_ENDIS_STAT)	R/W	0x0000_0000	See GTM Spec.(2)
0x08278 + CCi	TGC1 output enable control reg (TOM[i]_TGC1_OUTEN_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0827C + CCi	TGC1 output enable status reg (TOM[i]_TGC1_OUTEN_STAT)	R/W	0x0000_0000	See GTM Spec.(2)
(0x08030 – 0x0803C) + m*0x0040 + CCi	Reserved (m= 2..7, 10..15)			
(0x08400 – 0x087FC) + CCi	Reserved			
0x09000 – 0x0ABFC	Reserved for TOM2 to TOM5			
0x0AC00 – 0x0CFFC	Reserved			
ARU-connected Timer Output Module (ATOM) Registers (DDi = i*0x800, i = 0..2, m = 0..7)				
0x0D000 + m*0x0080 + DDi	ATOM Channel m ARU read address register (ATOM[i]_CH[m]_RDADDR)	R/W	0x01FE_01FE	See GTM Spec.(2)
0x0D004 + m*0x0080 + DDi	ATOM Channel m control register (ATOM[i]_CH[m]_CTRL)	R/W	0x0000_X000	See GTM Spec.(2)
0x0D008 + m*0x0080 + DDi	ATOM Channel m CCU0 compare shadow register (ATOM[i]_CH[m]_SR0)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D00C + m*0x0080 + DDi	ATOM Channel m CCU1 compare shadow register (ATOM[i]_CH[m]_SR1)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D010 + m*0x0080 + DDi	ATOM Channel m CCU0 compare register (ATOM[i]_CH[m]_CM0)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D014 + m*0x0080 + DDi	ATOM Channel m CCU1 compare register (ATOM[i]_CH[m]_CM1)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D018 + m*0x0080 + DDi	ATOM Channel m CCU0 counter register (ATOM[i]_CH[m]_CN0)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D01C + m*0x0080 + DDi	ATOM Channel m status register (ATOM[i]_CH[m]_STAT)	R/W	0x0000_0001	See GTM Spec.(2)

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x0D020 + m*0x0080 + DDi	ATOM channel m interrupt notification register (ATOM[i]_CH[m]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D024 + m*0x0080 + DDi	ATOM channel m interrupt enable register (ATOM[i]_CH[m]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D028 + m*0x0080 + DDi	ATOM channel m software interrupt generation (ATOM[i]_CH[m]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D02C + m*0x0080 + DDi	Channel m IRQ mode configuration register (ATOM[i]_CH[m]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
(0x0D030 – 0x0D03C) + m*0x0080 + DDi	Reserved for ATOM[i]_CH[m]			
0x0D040 + DDi	AGC Global control register (ATOM[i]_AGC_GLB_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D044 + DDi	AGC0 Enable/disable control register (ATOM[i]_AGC_ENDIS_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D048 + DDi	AGC Enable/disable status register (ATOM[i]_AGC_ENDIS_STAT)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D04C + DDi	AGC Action time base register (ATOM[i]_AGC_ACT_TB)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D050 + DDi	AGC Output enable control register (ATOM[i]_AGC_OUTEN_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D054 + DDi	AGC Output enable status register (ATOM[i]_AGC_OUTEN_STAT)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D058 + DDi	AGC Force update control register (ATOM[i]_AGC_FUPD_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D05C + DDi	AGC Internal trigger control register (ATOM[i]_AGC_INT_TRIG)	R/W	0x0000_0000	See GTM Spec.(2)
(0x0D040 – 0x0D05C) + m*0x0080 + DDi	Reserved for ATOM[i]_CH[m], m>1			
(0x0D060 – 0x0D07C) + m*0x0080 + DDi	Reserved for ATOM[i]_CH[m]			
(0x0D400 – 0x0D7FC) + DDi	Reserved for ATOM[i]			
0x0E800 – 0x12C40	Reserved for ATOM3 to ATOM11			
0x12C44 – 0x17FFC	Reserved			
F2A Configuration Registers (m = 0...7, i = 0, EEi = 0x4000)				
0x18000+m*0x04	F2A read channel address register (F2A[i]_CH[m]_ARU_RD_FIFO)	R/W	0x0000_01FE	See GTM Spec.(2)
0x18020+m*0x04	F2A stream x configuration register (F2A[i]_CH[m]_STR_CFG)	R/W	0x0000_0000	See GTM Spec.(2)

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x18040	F2A stream activation register (F2A[i]_ENABLE)	R/W	0x0000_0000	See GTM Spec.(2)
0x18044 – 0x1807C	Reserved			
AFD Registers (m = 0...7, i = 0, EEi = 0x4000)				
0x18080+m*0x10	AFD FIFO m buffer access register (AFD[i]_CH[m]_BUFF_ACC)	R/W	0x0000_0000	See GTM Spec.(2)
0x18084+m*0x10 – 0x1808C+m*0x10	Reserved for AFD			
0x18100 – 0x183FC	Reserved			
FIFO Configuration Registers (m = 0...7, i = 0)				
0x18400+m*0x40	FIFO Channel m control register (FIFO[i]_CH[m]_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x18404+m*0x40	FIFO Channel m end address register (FIFO[i]_CH[m]_END_ADDR)	R/W	0x0000_0XXX (3)	See GTM Spec.(2)
0x18408+m*0x40	FIFO Channel m start address register (FIFO[i]_CH[m]_START_ADDR)	R/W	0x0000_0XXX (3)	See GTM Spec.(2)
0x1840C+m*0x40	FIFO Channel m upper watermark register (FIFO[i]_CH[m]_UPPER_WM)	R/W	0x0000_0060	See GTM Spec.(2)
0x18410+m*0x40	FIFO Channel m lower watermark register (FIFO[i]_CH[m]_LOWER_WM)	R/W	0x0000_0020	See GTM Spec.(2)
0x18414+m*0x40	FIFO Channel x status register (FIFO[i]_CH[m]_STATUS)	R	0x0000_0005	See GTM Spec.(2)
0x18418+m*0x40	FIFO Channel m fill level register (FIFO[i]_CH[m]_FILL_LEVEL)	R	0x0000_0000	See GTM Spec.(2)
0x1841C+m*0x40	FIFO Channel m write pointer register (FIFO[i]_CH[m]_WR_PTR)	R	0x0000_0XXX (3)	See GTM Spec.(2)
0x18420+m*0x40	FIFO Channel m read pointer register (FIFO[i]_CH[m]_RD_PTR)	R	0x0000_0XXX (3)	See GTM Spec.(2)
0x18424+m*0x40	FIFO m interrupt notification register (FIFO[i]_CH[m]_IRQ_NOTIFY)	R/W	0x0000_0005	See GTM Spec.(2)
0x18428+m*0x40	FIFO m interrupt enable register (FIFO[i]_CH[m]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x1842C+m*0x40	FIFO m register to force interrupt by software (FIFO[i]_CH[m]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x18430+m*0x40	FIFO m IRQ mode control register (FIFO[i]_CH[m]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x18434+m*0x40	FIFO m error interrupt enable register (FIFO[i]_CH[m]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x18438+m*0x40 – 0x1843C+m*0x40	Reserved			
0x18600 – 0x18FFC	Reserved for FIFO0			
0x19000 – 0x19FFC	FIFO0 Memory			See GTM Spec.(2)
0x1A000 – 0x1BFFC	Reserved			
0x1C000 – 0x1DFFC	Reserved for F2A1, AFD1, FIFO1, and FIFO1 MEM			
0x1E000 – 0x27FFC	Reserved			
Digital PLL Module (DPLL)				
0x28000	Control Register 0 (DPLL_CTRL_0	R/W	0x003B_BA57	See GTM Spec.(2)
0x28004	Control Register 1 (DPLL_CTRL_1)	R/W	0xB000_0000	See GTM Spec.(2)
0x28008	Control Register 2 (actions 0–7 enable) (DPLL_CTRL_2)	R/W	0x0000_0000	See GTM Spec.(2)
0x2800C	Control Register 3 (actions 8–15 enable) (DPLL_CTRL_3)	R/W	0x0000_0000	See GTM Spec.(2)
0x28010	Control Register 4 (actions 16–24 enable) (DPLL_CTRL_4)	R/W	0x0000_0000	See GTM Spec.(2)
0x28014	Control Register 5 (actions 16–24 enable) (DPLL_CTRL_5)	R/W	0x0000_0000	See GTM Spec.(2)
0x28018	ACTION Status Register with connected shadow register (DPLL_ACT_STA)	R/W	0x0000_0000	See GTM Spec.(2)
0x2801C	Offset and switch old/new address register (DPLL_OSW)	R/W	0x0000_0200	See GTM Spec.(2)
0x28020	Address offset register for APT in RAM region 2 (DPLL_AOSV_2)	R/W	0x1810_0800	See GTM Spec.(2)
0x28024	Actual RAM pointer to RAM regions 2a, b and d (DPLL_APT)	R/W	0x0000_0000	See GTM Spec.(2)
0x28028	Actual RAM pointer to regions 1c1, 1c2 and 1c4 (DPLL_APS)	R/W	0x0000_0000	See GTM Spec.(2)
0x2802C	Actual RAM pointer to RAM region 2c (DPLL_APT_2c)	R/W	0x0000_0000	See GTM Spec.(2)
0x28030	Actual RAM pointer to RAM region 1c3 (DPLL_APS_1c3)	R/W	0x0000_0000	See GTM Spec.(2)
0x28034	Number of recent TRIGGER events used for calculations (mod 2*(TNU +1-SYN_NT)) (DPLL_NUTC)	R/W	0x0001_2001	See GTM Spec.(2)

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x28038	Number of recent STATE events used for calculations (mod 2*(SNU +1-SYN_NS) (DPLL_NUSC)	R/W	0x0000_2081	See GTM Spec.(2)
0x2803C	Number of active TRIGGER events to interrupt (DPLL_NTI_CNT)	R/W	0x0000_0000	See GTM Spec.(2)
0x28040	Interrupt notification register (DPLL_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x28044	Interrupt enable register (DPLL_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x28048	Interrupt force register (DPLL_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x2804C	Interrupt mode register (DPLL_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x28050	Error Interrupt enable register (DPLL_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x28054 – 0x280AC	Reserved			
0x280B0	Counter for pulses for TBU_TS1 to be sent in automatic end mode (DPLL_INC_CNT1)	R/W	0x0000_0000	See GTM Spec.(2)
0x280B4	Counter for pulses for TBU_TS2 to be sent in automatic end mode when SMC = RMO = 1w (DPLL_INC_CNT2)	R/W	0x0000_0000	See GTM Spec.(2)
0x280B8	Old RAM pointer and offset value for TRIGGER (DPLL_APT_SYNC)	R/W	0x0000_0000	See GTM Spec.(2)
0x280BC	Old RAM pointer and offset value for STATE (DPLL_APS_SYNC)	R/W	0x0000_0000	See GTM Spec.(2)
0x280C0	TBU_TS0 value at last TRIGGER event (DPLL_TBU_TS0_T)	R/W	0x0000_0000	See GTM Spec.(2)
0x280C4	TBU_TS0 value at last STATE event (DPLL_TBU_TS0_S)	R/W	0x0000_0000	See GTM Spec.(2)
0x280C8	Direct load input value for SUB_INC1 (DPLL_ADD_IN_LD1)	R/W	0x0000_0000	See GTM Spec.(2)
0x280CC	Direct load input value for SUB_INC2 (DPLL_ADD_IN_LD2)	R/W	0x0000_0000	See GTM Spec.(2)
0x280D0 – 0x280F8	Reserved for DPLL			
0x280FC	Status Register (DPLL_STATUS)	R/W	0x0000_0000	See GTM Spec.(2)
0x28100 + n*0x0004	9 bit ID information for input signal PMTR_n (8:0), n = 0...31 (DPLL_ID_PMTR_n)	R/W	0x0000_01FE	See GTM Spec.(2)
0x28180 – 0x281DC	Reserved for DPLL			
0x281E0	shadow register of DPLL_CTRL_0 (DPLL_CTRL_0_SHADOW_TRIGGER)	R/W	0x0000_0257	See GTM Spec.(2)

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value[1]	Location
0x281E4	shadow register of DPLL_CTRL_0 (DPLL_CTRL_0_SHADOW_STATE)	R/W	0x0000_0000	See GTM Spec.[2]
0x281E8	shadow register of DPLL_CTRL_1 (DPLL_CTRL_1_SHADOW_TRIGGER)	R/W	0x0000_0000	See GTM Spec.[2]
0x281EC	shadow register of DPLL_CTRL_1 (DPLL_CTRL_1_SHADOW_STATE)	R/W	0x0000_0000	See GTM Spec.[2]
0x281F0 – 0x281F8	Reserved			
0x281FC	Initialization Control and Status for RAMs (DPLL_RAM_INI)	R/W	0x0000_0000	See GTM Spec.[2]
DPLL RAM 1A Map				
	RAM Region 1a (0.288 KB for 96 words of 24 bits)			
0x28200 + n*0x0004	ACTION_n Position/Value action request Register (PSAn), n = 0...23	R/W	0x0000_0000	See GTM Spec.[2]
0x28260 + m*0x0004	ACTION_m time to react before PSAm (DLAm), m = 0...23	R/W	0x0000_0000	See GTM Spec.[2]
0x282C0 + n*0x0004	Number of TRIGGER/STATE increments to ACTION_n (NAn), n = 0...23	R/W	0x0000_0000	See GTM Spec.[2]
0x28320 + m*0x0004	Calculated relative time to ACTION_m (DTAm), m = 0...23	R/W	0x0000_0000	See GTM Spec.[2]
0x28380 – 0x283FC	Reserved on this RAM	R/W		
DPLL RAM 1B Map (Regions 1b and 1c)				
	DPLL RAM Region 1b (0.375 KB for 128 words of 24 bits) TRIGGER signal information stored **Note:** The switch of the LSB address bits is performed using the SWON register at 0x0020.			
0x28400/ 0x0404	Actual signal TRIGGER time stamp register TRIGGER_TS (TS_T)	R/W	0x0000_0000	See GTM Spec.[2]
0x28404/ 0x0400	Previous signal TRIGGER time stamp register TRIGGER_TS_old (TS_T_old)	R/W	0x0000_0000	See GTM Spec.[2]
0x28408	Actual signal TRIGGER filter and signal value (FTV_T)	R/W	0x0000_0000	See GTM Spec.[2]
0x2840C	Reserved[4]			
	STATE signal information stored **Note:** The switch of the LSB address bits is performed using the SWON register at 0x0020.	R/W	0x0000_0000	See GTM Spec.[2]
0x28410/ 0x0414	Actual signal STATE time stamp register STATE_TS (TS_S)	R/W	0x0000_0000	See GTM Spec.[2]
0x28414/ 0x0410	Previous signal STATE time stamp register STATE_TS_old (TS_S_old)	R/W	0x0000_0000	See GTM Spec.[2]
0x28418	Actual signal STATE filter and signal value (FTV_S)	R/W	0x0000_0000	See GTM Spec.[2]

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x2841C	Reserved(4)			
0x28420	TRIGGER hold time min value (THMI)	R/W	0x0000_0000	See GTM Spec.(2)
0x28424	TRIGGER hold time max value (THMA)	R/W	0x0000_0000	See GTM Spec.(2)
0x28428	Measured last pulse time from valid to invalid TRIGGER slope (THVAL)	R/W	0x0000_0000	See GTM Spec.(2)
0x2842C	Reserved(4)			
0x28430	Time out value of active TRIGGER slope (TOV)	R/W	0x0000_0000	See GTM Spec.(2)
0x28434	Time out value of active STATE slope (TOV_S)	R/W	0x0000_0000	See GTM Spec.(2)
0x28438	Calculated ADD_IN value for normal SUB_INC1 generation (ADD_IN_CAL1)	R/W	0x0000_0000	See GTM Spec.(2)
0x2843C	Calculated ADD_IN value for normal SUB_INC2 generation (ADD_IN_CAL2)	R/W	0x0000_0000	See GTM Spec.(2)
0x28440	missing pulses to be added/subtracted directly to SUB_INC1 and INC_CNT1 once (MPVAL1)	R/W	0x0000_0000	See GTM Spec.(2)
0x28444	missing pulses to be added/subtracted directly to SUB_INC2 and INC_CNT2 once (MPVAL2)	R/W	0x0000_0000	See GTM Spec.(2)
0x28448	target number of TRIGGER pulses (NMB_T_TAR)	R/W	0x0000_0000	See GTM Spec.(2)
0x2844C	target number of TRIGGER pulses (NMB_T_TAR_old)	R/W	0x0000_0000	See GTM Spec.(2)
0x28450	target number of STATE pulses (NMB_S_TAR)	R/W	0x0000_0000	See GTM Spec.(2)
0x28454	target number of STATE pulses (NMB_S_TAR_old)	R/W	0x0000_0000	See GTM Spec.(2)
0x28458 – 0x2845C	Reserved(4)			
0x28460	reciprocal value of expected increment duration (T) (RCDT_TX)	R/W	0x0000_0000	See GTM Spec.(2)
0x28464	reciprocal value of expected increment duration (S) (RCDT_SX)	R/W	0x0000_0000	See GTM Spec.(2)
0x28468	reciprocal value of the expected nominal increment duration (T) (RCDT_TX_nom)	R/W	0x0000_0000	See GTM Spec.(2)
0x2846C	reciprocal value of the expected nominal increment duration (S) (RCDT_SX_nom)	R/W	0x0000_0000	See GTM Spec.(2)
0x28470	actual reciprocal value of TRIGGER (RDT_T_actual)	R/W	0x0000_0000	See GTM Spec.(2)

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value[1]	Location
0x28474	actual reciprocal value of STATE (RDT_S_actual)	R/W	0x0000_0000	See GTM Spec.[2]
0x28478	Duration of last TRIGGER increment (DT_T_actual)	R/W	0x0000_0000	See GTM Spec.[2]
0x2847C	Duration of last STATE increment (DT_S_actual)	R/W	0x0000_0000	See GTM Spec.[2]
0x28480	Absolute error of prediction for last TRIGGER increment (EDT_T)	R	0x0000_0000	See GTM Spec.[2]
0x28484	Average absolute error of prediction up to the last TRIGGER increment (MEDT_T)	R	0x0000_0000	See GTM Spec.[2]
0x28488	absolute error of prediction for last STATE increment (EDT_S)	R	0x0000_0000	See GTM Spec.[2]
0x2848C	Average absolute error of prediction up to the last STATE increment (MEDT_S)	R	0x0000_0000	See GTM Spec.[2]
0x28490	Expected duration of current TRIGGER increment (CDT_TX)	R	0x0000_0000	See GTM Spec.[2]
0x28494	Expected duration of current STATE increment (CDT_SX)	R	0x0000_0000	See GTM Spec.[2]
0x28498	Expected nominal duration of current TRIGGER increment (without consideration of missing events) (CDT_TX_nom)	R	0x0000_0000	See GTM Spec.[2]
0x2849C	Expected nominal duration of current STATE increment (without consideration of missing events) (CDT_SX_nom)	R	0x0000_0000	See GTM Spec.[2]
0x284A0	TRIGGER Locking Range (TLR)	R/W	0x0000_0000	See GTM Spec.[2]
0x284A4	STATE Locking Range (SLR)	R/W	0x0000_0000	See GTM Spec.[2]
0x28500 + n*0x0004	For ACTION_n in normal mode (PDT_n), n = 0...31	R/W	0x0000_0000	See GTM Spec.[2]
0x28580 – 0x285BC	Reserved[4]			
0x285C0	Calculated number of sub-pulses between two STATE events (to be set by CPU) (MLS1)	R/W	0x0000_0000	See GTM Spec.[2]
0x285C4	Calculated number of sub-pulses between two STATE events (to be set by CPU) for the use when SMC = RMO = 1B (MLS2)	R/W	0x0000_0000	See GTM Spec.[2]
0x285C8	number of sub-pulses of SUB_INC1 in continuous mode, updated by the host only (CNT_NUM1)	R/W	0x0000_0000	See GTM Spec.[2]
0x285CC	number of sub-pulses of SUB_INC2 in continuous mode, updated by the host only (CNT_NUM2)	R/W	0x0000_0000	See GTM Spec.[2]

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x285D0	Plausibility value of next active TRIGGER slope (PVT)	R/W	0x0000_0000	See GTM Spec.(2)
0x285E0	Accurate calculated position stamp of last TRIGGER input (PSTC)	R/W	0x0000_0000	See GTM Spec.(2)
0x285E4	Accurate calculated position stamp of last STATE input (PSSC)	R/W	0x0000_0000	See GTM Spec.(2)
0x285E8/ 0x285EC	Measured position stamp of last TRIGGER input (PSTM)	R/W	0x0000_0000	See GTM Spec.(2)
0x285EC/ 0x285E8	Measured position stamp of last but one TRIGGER input (PSTM_old)	R/W	0x0000_0000	See GTM Spec.(2)
0x285F0/ 0x285F4	Measured position stamp of last STATE input (PSSM)	R/W	0x0000_0000	See GTM Spec.(2)
0x285F4/ 0x285F0	Measured position stamp of last but one STATE input (PSSM_old)	R/W	0x0000_0000	See GTM Spec.(2)
0x285F8	Number of pulses of current increment in normal mode for SUB_INC1 (NMB_T)	R/W	0x0000_0000	See GTM Spec.(2)
0x285FC	Number of pulses of current increment in emergency mode for SUB_INC1 (NMB_S)	R/W	0x0000_0000	See GTM Spec.(2)
	DPLL RAM Region 1c (0.75 KB for 256 words of 24 bits) **Note:** The following registers for the signal STATE are located in RAM Region 1c, read access by AEI via bus interface possible, writing results in an interrupt; data width of 3 bytes used for 24-bit values.			
	1c1			
0x28600 + n*0x0004	Nominal STATE reciprocal values in FULL_SCALE (RDT_Sn), n = 0...63 (2*(SNU+1-SYN_NS) valid entries)	R/W	0x0000_0000	See GTM Spec.(2)
	1c2			
0x28700 + m*0x0004	Time stamp field of STATE Events (TSF_Sm), m = 0...63 (2*(SNU+1) valid entries)	R/W	0x0000_0000	See GTM Spec.(2)
	1c3			
0x28800 + n*0x0004	Adapt Values for all STATE Increments (ADT_Sn), n = 0...63 (2*(SNU+1-SYN_NS) valid entries)	R/W	0x0000_0000	See GTM Spec.(2)
	1c4			
0x28900 + m*0x0004	Nominal STATE Increment Values for FULL_SCALE (DT_Sm), m = 0...63 (2*(SNU+1-SYN_NS) valid entries)	R	0x0000_0000	See GTM Spec.(2)
0x28A00 – 0x28DFC	Reserved			

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x28E00 + n*0x0004	Calculated ACTION_n TIME STAMP (DPLL_TSACn), n = 0...31	R/W	0x0000_0000	See GTM Spec.(2)
0x28E80 + n*0x0004	Calculated position value for action n (DPLL_PSACn), n = 0...31	R/W	0x0000_0000	See GTM Spec.(2)
0x28F00	control bits for actions 0...3 (DPLL_ACB_0)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F04	control bits for actions 4...7 (DPLL_ACB_1)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F08	control bits for actions 8....11 (DPLL_ACB_2)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F0C	control bits for actions 12...15 (DPLL_ACB_3)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F10	control bits for actions 16...19 (DPLL_ACB_4)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F14	control bits for actions 20...23 (DPLL_ACB_5)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F18	control bits for actions 24...27 (DPLL_ACB_6)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F1C	control bits for actions 28...31 (DPLL_ACB_7)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F20 – 0x2BFFC	Reserved			
DPLL RAM 2 map				
	Region 2a			
0x2C000 + n*0x0004	TRIGGER nominal increment reciprocals in FULL_SCALE (RDT_Tn), n = 0 ... N-1, with N = 2*(TNU+1-SYN_NT) valid entries, and N = 128...1024 OBS: final address 2a = 0x2C000 + (N-1)*0x4	R/W	0x0000_0000	See GTM Spec.(2)
	Region 2b			
final address 2a + 0x4 + n*0x0004	Time Stamp Field of TRIGGER events (TSF_Tn), n = 0...N-1, with N = 2*(TNU+1) valid entries, and N = 128...1024 OBS: the final address 2b depends on N	R/W	0x0000_0000	See GTM Spec.(2)
	Region 2c			
final address 2b + 0x4 + m*0x0004	Adapt Values for all Increments (ADT_Tm), m = 0...M-1, with M = 2*(TNU+1-SYN_NT) valid entries, and M = 128...1024 OBS: the final address 2c depends on M	R/W	0x0000_0000	See GTM Spec.(2)
	Region 2d			

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
final address 2c + 0x4 + m*0x0004	nominal TRIGGER increment values for FULL_SCALE (DT_Tm), m = 0...M-1, with M = 2*(TNU+1-SYN_NT) valid entries, and M = 128...1024 OBS: the final address 2d depends on M	R/W	0x0000_0000	See GTM Spec.(2)
Multi Channel Sequencer (MCS) Configuration Registers (FFi = i*0x1000, i = 0..1, n = 0..7)				
0x30000+n*0x80+ y*0x04 + FFi	MCS Channel n GPR registers (MCS[i]_CH[n]_R[y]), y = 0...7	R/W	0x0000_0000	See GTM Spec.(2)
0x30020+n*0x80 + FFi	MCS Channel n control register (MCS[i]_CH[n]_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x30024+n*0x80 + FFi	MCS Channel n ACB register (MCS[i]_CH[n]_ACB)	R	0x0000_0000	See GTM Spec.(2)
0x30028 + FFi	MCS Clear trigger control register (MCS[i]_CTRG)	R/W	0x0000_0000	See GTM Spec.(2)
0x3002C + FFi	MCS Set trigger control register (MCS[i]_STRG)	R/W	0x0000_0000	See GTM Spec.(2)
(0x30028 – 0x3002C) + n*0x80 + FFi	Reserved (n > 0)			
(0x30030 – 0x3003C) + n*0x80 + FFi	Reserved			
0x30040+n*0x80 + FFi	MCS Channel n Program counter register (MCS[i]_CH[n]_PC)	R/W	0x0000_0000 + 4*n	See GTM Spec.(2)
0x30044+x*0x80 + FFi	MCS Channel n interrupt notification register (MCS[i]_CH[n]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x30048+n*0x80 + FFi	MCS Channel n interrupt enable register (MCS[i]_CH[n]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x3004C+n*0x80 + FFi	MCS Channel n software interrupt generation register (MCS[i]_CH[n]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x30050+n*0x80 + FFi	IRQ mode configuration register (MCS[i]_CH[n]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x30054+n*0x80 + FFi	MCS Channel n error interrupt enable register (MCS[i]_CH[n]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
(0x30058 – 0x30070) + n*0x80 + FFi	Reserved			
0x30074 + FFi	MCS Control register (MCS[i]_CTRL)	R/W	0x000X_0000	See GTM Spec.(2)
0x30078 + FFi	MCS Channel reset register (MCS[i]_RST)	R/W	0x0000_0000	See GTM Spec.(2)

Table 537. GTMINT module memory map(Continued)

Offset	Use	Access	Reset value(1)	Location
0x3007C + FFi	MCS Error register (MCS[i]_ERR)	R/W	0x0000_0000	See GTM Spec.(2)
(0x30074 – 0x3007C) + n*0x80 + FFi	Reserved (n > 0)			
(0x30400 – 0x30FFC) + FFi	Reserved			
0x32000 – 0x37FFC	Reserved for MCS2 / 3 / 4 / 5 / 6 registers			
MCS Memories (GGi = i*0x8000, i = 0..1)				
(0x38000 – 0x3BFFC) + GGi	MCS Memory (MCS[i]_MEM)			See GTM Spec.(2)
(0x3C000 – 0x3FFFC) + GGi	Reserved			
0x48000 – 0x6BFFC	Reserved for MCS2 / 3 / 4 / 5 / 6 Memories			
0x6C000 – 0xFFFFC	Reserved			

1. Default value when in reset state.
2. For detailed information see the *GTM-IP Specification* document.
3. Reset value depends on configuration parameters.
4. RAM reserved position for GTM-IP internal use. No error occurs when accessed.

43.2.2 Register descriptions

This section contains the description of additional registers on the top integration module. Description of GTM-IP registers, or GTMDI registers are found in their respective specification document.

43.2.2.1 GTM Module Configuration Register (GTMMCR)

The GTMMCR contains the control bits to configure the general operation of the GTMINT and GTM-IP Module.

Offset 0x000C0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MDIS	0	0	0	0	0	0	0	0	0	0	0	0	0	AEISREN
W																
Reset	0	—(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	STPS	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	—(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 478. GTM Module Configuration Register (GTMMCR)

1. Reset value is SoC dependent.

GTMMCR register functions are as shown in *Table 538*.

Table 538. GTMMCR field descriptions

Field	Description
1 MDIS	MDIS Module Disable. The MDIS bit puts the GTMINT Module in stop/disabled mode. Communication through the slave-bus Interface is ignored in this mode except reads/writes to the GTMMCR and reads from other integration registers which are allowed. 1 Module disable request 0 Normal Mode
15 AEISREN	AEISREN AEI interface soft-reset control enable. The AEISREN bit enables the soft-reset of the AEI interface that is controlled by AEISRST bit in the GTMMAEICR. Refer to *Section 43.3.7, AEI Interface Software Reset Description* for more details. 1 AEI soft-reset control is enabled 0 AEI soft-reset control is disabled
17 STPS	STPS Stop Mode Status. The STPS bit indicates the GTM-IP Module is in stop mode, or by MDIS bit field setting or by stop line request. Refer to *Section 43.3.4, Stop Mode Description* for more details. 1 GTM is in Stop Mode 0 Normal Mode

43.2.2.2 GTM Interrupts Clear Register (GTMINTCLR)

The GTMINTCLR contains the control bits to clear some specific interrupt request output.

Offset 0x000C8 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	0
W							INTCLR_PTR[0:9]									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 479. GTM Interrupts Clear Register (GTMINTCLR)

GTMINTCLR register functions are as shown in *Table 539*.

Table 539. GTMINTCLR field descriptions

Field	Description
22–31 INTCLR_PTR[0:9]	INTCLR_PTR Interrupt Clear Pointer. This field value indicates / selects which interrupt will be cleared. The correspondence between the number and the interrupt is given in *Table 540*. Refer to *Section 43.3.8, Interrupt and DMA Support Logic* for more details.

Table 540. Correspondence between the INTCLR_PTR and the interrupts outputs

INTCLR_PTR[0:9]		Interrupt output signal to be cleared
HEX	DEC	
0x000	0	No interrupt is selected
0x001 to 0x2C1	1 to 705	Reserved
0x2C2	706	ipi_int_gtm_aei
0x2C3 to 0x2C5	707 to 709	ipi_int_gtm_aru[0]–[2]
0x2C6	710	Reserved
0x2C7	711	ipi_int_gtm_cmp
0x2C8	712	ipi_int_gtm_spe0
0x2C9	713	ipi_int_gtm_spe1
0x2CA to 0x2D1	714 to 721	ipi_int_gtm_psm0[0]–[7]
0x2D2 to 0x2EC	722 to 748	ipi_int_gtm_dpll[0]–[26]
0x2ED to 0x2F4	749 to 756	ipi_int_gtm_tim0[0]–[7]
0x2F5 to 0x2FC	757 to 764	ipi_int_gtm_tim1[0]–[7]
0x2FD to 0x30C	765 to 780	Reserved
0x30D to 0x314	781 to 788	ipi_int_gtm_mcs0[0]–[7]
0x315 to 0x31C	789 to 796	ipi_int_gtm_mcs1[0]–[7]
0x31D to 0x32C	797 to 812	Reserved
0x32D to 0x334	813 to 820	ipi_int_gtm_tom0[0]–[7]

Table 540. Correspondence between the INTCLR_PTR and the interrupts outputs(Continued)

INTCLR_PTR[0:9]		Interrupt output signal to be cleared
HEX	DEC	
0x335 to 0x33C	821 to 828	ipi_int_gtm_tom1[0]–[7]
0x33D to 0x344	829 to 836	Reserved
0x345 to 0x348	837 to 840	ipi_int_gtm_atom0[0]–[3]
0x349 to 0x34C	841 to 844	ipi_int_gtm_atom1[0]–[3]
0x34D to 0x350	845 to 848	ipi_int_gtm_atom2[0]–[3]
0x351 to 0x3A2	849 to 930	Reserved
0x3A3	931	ipi_int_gtm_err
0x3A4 to 0x3FF	932 to 1023	Reserved

43.2.2.3 GTM AEI Control Register (GTMMAEICR)

The GTMMAEICR contains the control bits to the AEI interface in GTM-IP.

Offset 0x000CC 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	0
W																AEISRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 480. GTM AEI Control Register (GTMMAEICR)

GTMMAEICR register functions are as shown in *Table 541*.

Table 541. GTMMAEICR field descriptions

Field	Description
31 AEISRST	AEISRST AEI interface soft-reset control. The AEISRST bit when set applies a reset on the GTM-IP AEI interface. The set is enabled by the AEISREN bit in the GTMMCR. This is a self clear bit and is always read as 0. Refer to *Section 43.3.7, AEI Interface Software Reset Description* for more details.

43.3 Functional description

The following sections describe the GTMINT Module functionality.

The GTMINT Module is designed to accommodate the multipurpose timer (GTM-IP) and the debugger block (GTMDI). The GTM-IP can generate timed signals and also can handle external signals to obtain their timing information. The GTMDI provides several features to help the development of programs to be run on GTM-IP. Refer to *Figure 477* for the integration architecture.

The GTMINT Module integrates the following internal submodules into the SPC572Lx platform:

- GTM-IP timer
- GTMDI debug interface
- ECC error correction sub-blocks
- Interface match logic

The following sections describe each one of these submodules and their combined operation.

43.3.1 GTM-IP Timer

The Generic Timer Module (GTM) is composed of several submodules of different functionality. These submodules can be combined in a configurable manner to form a complex timer module that serves different application domains and different classes within one application domain. Because of this scalability and configurability the timer is called generic. The scalability and configurability is reached with an architecture philosophy where dedicated hardware submodules are located around a central routing unit called Advanced Routing Unit (ARU). The ARU can connect the submodules in a flexible manner. The connectivity is software programmable and can be configured during runtime.

Nevertheless, the GTM-IP is designed to unload the CPU or a peripheral core from a high interrupt load. Most of the tasks inside the GTM-IP can run (once set up by an external CPU) independent and in parallel to the software. There may be special situations, where the CPU has to take action but the goal of the GTM design was to reduce these situations to a minimum. In addition, the timer section of GTM-IP runs at a fixed clock frequency (80 MHz) to allow real time period counting, but the interface to CPU can work with a frequency variable clock.

43.3.1.1 GTM-IP Timer Architecture

The hardware submodules have dedicated functions, e.g., there are timer input modules where incoming signals can be captured and characterized together with a notion of time. By combination of several submodules through the ARU complex functions can be established. For example, the signals characterized at an input module can be routed to a signal processing unit where an intermediate value about the incoming signal frequency can be calculated.

The modules that help to implement such complex functions are called infrastructural components further on. These components are present in all GTM variants. However, the number of these components may vary from device to device. Other submodules have a more general architecture and can fulfill typical timer functions, e.g., there are PWM generation units. The third class of submodules are those fulfilling a dedicated functionality for a certain application domain, e.g., the DPLL serves engine management applications. A fourth group of submodules is responsible for supporting the implementation of safety functions to fulfill a defined safety level.

Each GTM-IP is build up therefore with submodules coming from those four groups. The application class is defined by the amount of components of those submodules integrated into the implemented GTM-IP.

A list of submodules is given in *Table 542*.

Table 542. Submodules of GTM-IP

Submodule	Group
Advanced Routing Unit (ARU)	Infrastructural components
First In First Out Module (FIFO)	
AEI-to-FIFO Data Interface (AFD)	
FIFO-to-ARU Interface (F2A)	
Clock Management Unit (CMU)	
Time Base Unit (TBU)	
Timer Input Module (TIM)	IO modules
Timer Output Module (TOM)	
ARU-connected Timer Output Module (ATOM)	
Multi Channel Sequencer (MCS)	Signal generation and processing
Memory Configuration Module (MCFG)	Infrastructural component for MCS
TIM0 Input Mapping Module (MAP)	Dedicated
Digital PLL (DPLL)	
Sensor Pattern Evaluation Module (SPE)	BLDC support
Interrupt Concentrator Module (ICM)	Interrupt services
Output Compare Unit (CMP)	Safety features
Monitoring Unit (MON)	

For more details, refer to the *GTM-IP Specification* document.

43.3.2 GTMDI Debug Interface

The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including MCS cores, ARU data trace, Input Channel Filters trace, Output Timer channels, DPLL and SPE submodules. The GTMDI interfaces to the GTM IP using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules in order to implement complex debug features. Trace capability is also provided along with a global time-stamp which allows real time trace for several GTM submodule events.

The GTMDI implements a Nexus Client standard interface and a JTAG standard Port Controller to provide communication capability with the JTAG and Nexus Aurora router on-chip modules. Note that the Nexus standard is only used for the physical interfaces. In order to implement the debug features such as breakpoint and messages required for the GTM debug, the Nexus standard could not be followed.

For detailed architecture and features description, refer to the GTMDI chapter of this reference manual.

43.3.2.1 Peripheral Slave Bus Read Debug Access

Debugger is able to read all GTM registers, which are visible from the peripheral bus even when the timer is running. Read-access of the debugger is possible without destroying or changing the contents of these registers. A read debug access using the slave bus has no side effects on the timer. The main example is a read access to the FIFO that does not update the FIFO pointers.

43.3.3 Error Detection on GTM Memories Accesses

The GTM-IP architecture requires 8 RAM units that should be placed on the SoC. Each RAM is connected with separated input and output ports to the GTM-IP for safety purposes.

Memories of this block are equipped with error detection features for safety improvement. The following procedures detect errors on memory accesses:

- Memory write data and its address are considered in the parity bits calculation. These parity bits are stored in that memory address as part of the written data.
- Memory read data and its address are considered in the error detection. The parity bits are part of the data read from the specified memory address.
- Error Correcting Code (ECC), that corrects single errors and detects double errors on each read access

43.3.3.1 ECC Description

The Error Correcting Code (ECC) for GTM RAMs provides guaranteed single bit error correction and double bit error detection (without correction) per accessed word. On write accesses, the parity bits are calculated and appended to the original data bits to be written into the specified memory address. And error detection and correction are performed on read accesses.

Note: In case of error detection (with correction or not), no write operation is done here to fix possible errors in the memory content.

Note: All memories must be initialized with write operations in all addresses because the ECC parity bits need to be stored in each memory address before any read operation.

Note: GTM-IP sub-block does this initialization automatically for all memories, except for FIFO0 RAM. In this case, the user must be careful to execute writes before reads.

Table 543 presents a summary of GTM memories and the necessary ECCs for single error correction / double error detection.

Table 543. ECC characteristics for GTM memories

Memory	MEM number of words	MEM address width (bits)	Uncoded data width (bits)	ECC (bits)	MEM data in/out (data + ECC) (bits)	Matrix size (number of syndrome results)
FIFO0 RAM	1024	10	29	7	36	46
DPLL RAM1A	96	7	24	7	31	38
DPLL RAM1B	384	9	24	7	31	40
DPLL RAM2	512	9	24	7	31	40
MCS0 RAM0	1024	10	32	7	39	49

Table 543. ECC characteristics for GTM memories(Continued)

Memory	MEM number of words	MEM address width (bits)	Uncoded data width (bits)	ECC (bits)	MEM data in/out (data + ECC) (bits)	Matrix size (number of syndrome results)
MCS1 RAM0	1024	10	32	7	39	49
MCS0 RAM1	512	9	32	7	39	48
MCS1 RAM1	512	9	32	7	39	48

43.3.3.1.1 Error Correcting Code (ECC) and Syndrome Definition

This ECC is based on the Hsiao code with 7 parity bits for each data word.

Table 544 shows the definition of the syndrome field for each bit position. Note that the ECC data vector width can be different for each memory, so we consider only the necessary positions for each one, not all the table positions.

Table 544. 42 bits Word Based Syndrome Definition

SYND[6:0] / Element value (hex)	Error result (for SMATRIX)	Matrix element (for MATRIX)	SYND[6:0] / Element value (hex)	Error result (for SMATRIX)	Matrix element (for MATRIX)
40	error on **code** bit 6	—	54	error on data bit 23	element 23
20	error on **code** bit 5	—	15	error on data bit 22	element 22
10	error on **code** bit 4	—	16	error on data bit 21	element 21
08	error on **code** bit 3	—	34	error on data bit 20	element 20
04	error on **code** bit 2	—	25	error on data bit 19	element 19
02	error on **code** bit 1	—	26	error on data bit 18	element 18
01	error on **code** bit 0	—	64	error on data bit 17	element 17
4F	error on data(1) bit 41	element 41	2C	error on data bit 16	element 16
3E	error on data bit 40	element 40	1A	error on data bit 15	element 15
3D	error on data bit 39	element 39	23	error on data bit 14	element 14
3B	error on data bit 38	element 38	2A	error on data bit 13	element 13
37	error on data bit 37	element 37	32	error on data bit 12	element 12
2F	error on data bit 36	element 36	46	error on data bit 11	element 11
1F	error on data bit 35	element 35	4A	error on data bit 10	element 10
70	error on data bit 34	element 34	52	error on data bit 9	element 9
68	error on data bit 33	element 33	62	error on data bit 8	element 8
07	error on data bit 32	element 32	13	error on data bit 7	element 7
49	error on data bit 31	element 31	29	error on data bit 6	element 6
0D	error on data bit 30	element 30	31	error on data bit 5	element 5
0E	error on data bit 29	element 29	43	error on data bit 4	element 4

Table 544. 42 bits Word Based Syndrome Definition(Continued)

SYND[6:0] / Element value (hex)	Error result (for SMATRIX)	Matrix element (for MATRIX)	SYND[6:0] / Element value (hex)	Error result (for SMATRIX)	Matrix element (for MATRIX)
38	error on data bit 28	element 28	45	error on data bit 3	element 3
4C	error on data bit 27	element 27	19	error on data bit 2	element 2
1C	error on data bit 26	element 26	51	error on data bit 1	element 1
58	error on data bit 25	element 25	61	error on data bit 0	element 0
0B	error on data bit 24	element 24	—	—	—
—	—	—	—	—	—
00	no error	—	any other(2)	non-correctable error	—

1. Actually the data can be data or address bit.
2. For the FIFO0 RAM with 29 data bits and 10 address bits, the valid error positions are from bit 0 to bit 38, only. The syndromes with values for bit 39 to 41 of the table are considered also for non-correctable errors.

43.3.3.2 Error Report Description

Each RAM generates a defined number of information to report an error on an access. Each of them is described below.

- Error Address – this is the mapped address of the RAM as defined in *Section 43.2.1, Module memory map*. This field has only meaning when an error occurs and is flagged by the two error signals below. It is reported on read or write operations.
- Non-correctable Error – This one bit width error indicates that:
  - The ECC detected errors in the read access and it is not possible to correct them because the ECC can only fix single bit errors. The error can occur in the RAM read data bus (parity bits + data) or in the address value that the RAM has received.
- Correctable Error – this one bit width error indicates that:
  - The ECC detected and corrected a single bit error on the RAM read data bus (parity bits + data). The parity bits are discarded after the ECC decoder but this error report includes these bits to obtain the performance of the whole RAM data storage sector.

This Error Report remains active for only one RAM clock cycle and should be captured by the external RAM Error Monitor module. These error indications are also available for the GTM-IP timer module that is the block that requests the RAM accesses. Uncorrectable ECC error indication are provided to GTM-IP but the reaction to the error depends on this block itself. Refer to its specification for more details.

43.3.4 Stop Mode Description

The stop mode is characterized by the stopping of the functional clock and the timer configuration registers clock. Therefore, the timers become stopped and almost all registers access are not allowed. This mode can be requested by two ways:

- By asserting field GTMMCR[MDIS].
- By receiving the SoC stop request signal ipg_stop.
  - the minimum waiting time to a new stop request is about 20 bus clock cycles.

However, it is important to note that to confirm the stop request was granted, it is necessary to read the stop status field GTMMCR[STPS].

43.3.4.1 Stop Mode by MDIS

The stop mode is requested by asserting this bit field with a write operation on the GTMMCR mapped register. The module only requests to stop the clocks when no slave bus accesses are pending in the GTM-IP. When in stop mode:

- The clocks are disabled to feed the whole GTM-IP and peripheral functions like the ECC
- The stop status field GTMMCR[STPS] is set.
- Read or write accesses to all mapped registers are not allowed and generates an access error indication, except accesses for the GTMMCR.
- The slave bus clock is not affected by stop mode.

This stop mode ends by clearing GTMMCR[MDIS]. GTMMCR[STPS] is automatically cleared (it is considered that the SoC stop request is not set).

43.3.5 Debug / Halt Mode Description

The debug / halt mode is requested via GTMDI and is used to debug the GTM-IP sub-block. In this mode it is required that the core of GTM-IP becomes halted by stopping its clock. Therefore all time bases and counters are frozen.

Debug or halt mode requests should be enabled by setting the bit field DBE on the GTMDI GTMDI_DC register that is accessible by the JTAG interface (refer to the GTMDI chapter for more details). GTM halt is only activated when no read- or write-access to GTM over AEI are taking place in parallel. Only in this condition the GTM is released to change to debug / halt mode read.

It is important to notice that any slave bus access is active in this mode. Debugger is able to read all GTM addresses read-/writeable from the peripheral bus when GTM is running or halted. All registers in the register map can be accessed by the slave bus, even the RAMs because they are in the register map. Of course, the clocks for the RAMs are not stopped in debug mode.

For more details about the debug mode, refer to GTMDI documentation.

Note: If this module is in disable mode by setting GTMMCR[MDIS], the debug request is not considered until the MDIS bit is cleared.

43.3.6 GTM-IP Timer Software Reset Description

The GTM-IP timer portion has the software reset possibility by writing a ‘1’ in the RST bit field on the GTM_RST register. This bit is self cleared and is always read as zero. In addition, this bit is write protected by the bit field RF_PROT in the GTM_CTRL register.

This is a global reset to GTM-IP block only. The GTMDI is not affected by this soft-reset.

43.3.7 AEI Interface Software Reset Description

The GTM-IP has software reset control for the AEI interface sub-block only. This block is the slave bus interface of GTM-IP and is not reset by the GTM-IP timer soft-reset. The AEI soft-reset occurs by writing a ‘1’ in the field GTMMAEICR[AEISRST]. This bit is self cleared and

is always read as zero. In addition, this bit is write protected by the field GTMMCR[AEISREN].

A soft-reset in the AEI interface is equivalent to the hard-reset. It occurs in the next clock cycle of the request and takes one bus clock cycle. The AEI buffer and configuration registers are cleared, and some slave bus access can be aborted. Therefore, access commands in the buffer memory are lost and the configuration registers of the AEI interface change to their reset values.

43.3.8 Interrupt and DMA Support Logic

The GTM-IP provides several interrupts / DMA request signals to signalize internal events to other external modules. The list of all signals are provided in the GTM-IP documentation and all have the same type of controls and configuration options.

Each interrupt / DMA request line represents a set of internal interrupt / DMA request signals, and each of this internal signal has its own set of status / control / configuration registers: IRQ_EN, IRQ_NOTIFY, IRQ_FORCINT, and IRQ_MODE registers. Refer to *GTM-IP specification* for more details.

It is required to maintain the configuration register IRQ_MODE = 00 that is the Level Interrupt Mode, the default mode to be compatible with the chip standard. the IRQ_NOTIFY register is the internal flag and can be enabled to set an external signal by the IRQ_EN register. For more details, refer to the GTM-IP documentation.

The GTM integration module provides the same GTM-IP set of signals as interrupt requests and as DMA requests and the real functionality on the chip can be configured later. However, if a signal is configured to start a DMA request, the request can be cleared by the respective DMA answer, but it can be also cleared by software as occurs for interrupt requests.

43.3.8.1 Interrupt Request Clear Controls

Each interrupt request output can be a composition of several internal events flags that are enabled by configuration registers. After an interrupt request is recognized by the chip interrupt controller and it is serviced, the interrupt routine must clear the flags that originated that service request. The GTM integration module provides two ways to clear those flags, but each method has its limitations that are described below.

The first way to clear the events flags that originated the interrupt request is by a slave bus write operation in the interrupt mapped registers named *_IRQ_NOTIFY in *Table 537*. The main characteristics are listed below:

- For example, writing a ‘1’ in the bit field AEI_IM_ADDR on the register GTM_IRQ_NOTIFY is going to clear this bit to clear the ipi_int_gtm_aei interrupt output signal
- Several flags in this register could have generated the interrupt request, therefore all of them can be cleared with this write operation.
- There is a delay between the reception of the clear command and the actual clear of the flags. This time interval can vary and depends on the number of commands inside the AEI buffer. Therefore it is not possible to know easily when the flags were cleared.
- This method of interrupt clear can generate a new service request if we are using write masked response mode for AEI interface.

The second way to clear the events flags of the interrupt requests is by using the mapped register GTMINTCLR that is described in *Section 43.2.2.2, GTM Interrupts Clear Register (GTMINTCLR)*. The main characteristics of this method are listed below:

- For example, to clear the ipi_int_gtm_aei interrupt output signal, we execute a write operation in the GTMINTCLR with data value 0x2C2 in the INTCLR_PTR bit field. This value selects to execute a clear in all flags that are used to generate that interrupt request.
- There is a delay between the reception of the clear command and the actual clear of the flags. This time interval can vary and depends on the number of commands inside the AEI buffer. However, the flags clear instant is well defined and can be seem by the integration module.
- As this method clear all flags for an interrupt request output, some flag that is set during the interrupt service is also cleared, and then some event can be lost.

43.4 Initialization/Application Information

The GTM integration module receives two active low reset signals, one on the slave bus clock domain, and the other in the peripheral clock domain. The last one resets all the timer logic, and registers inside the GTM-IP block. Just after this reset signal is negated, the GTM-IP is in the following condition:

- The GTM bridge is in asynchronous mode.
- after reset, write operations waits the response status (write response is not masked).
- If MDIS_DEFAULT is ‘1’, a write access to GTMMCR is necessary to clear the MDIS bit register and have the GTM-IP clocks toggling.
- DPLL RAMs and MCSs RAMs starts the write initialization cycles. This is necessary for ECC parity bits storage on each RAM address.
- FIFO0 RAM is not automatically initialized. In normal operation mode, the initialization is not required because the FIFO always have to write the memory first to read later.

Refer to GTM-IP and GTMDI documentation to initialize the GTMINT module.

44 CAN Subsystem

44.1 Introduction

The Controller Area Network (CAN) subsystem consists of the modular CAN (M_CAN) modules with an integrated intelligent CAN RAM controller. The CAN RAM controller consists of additional logic for arbitration between the requests for the RAM access by the various CANs and CPU, ECC encoder/decoder for the Message RAM data and active transmit message buffer protection from CPU write access. The subsystem follows the little endian format.

Note: Refer to the Device Configuration chapter to see the number of CAN nodes used in the device.

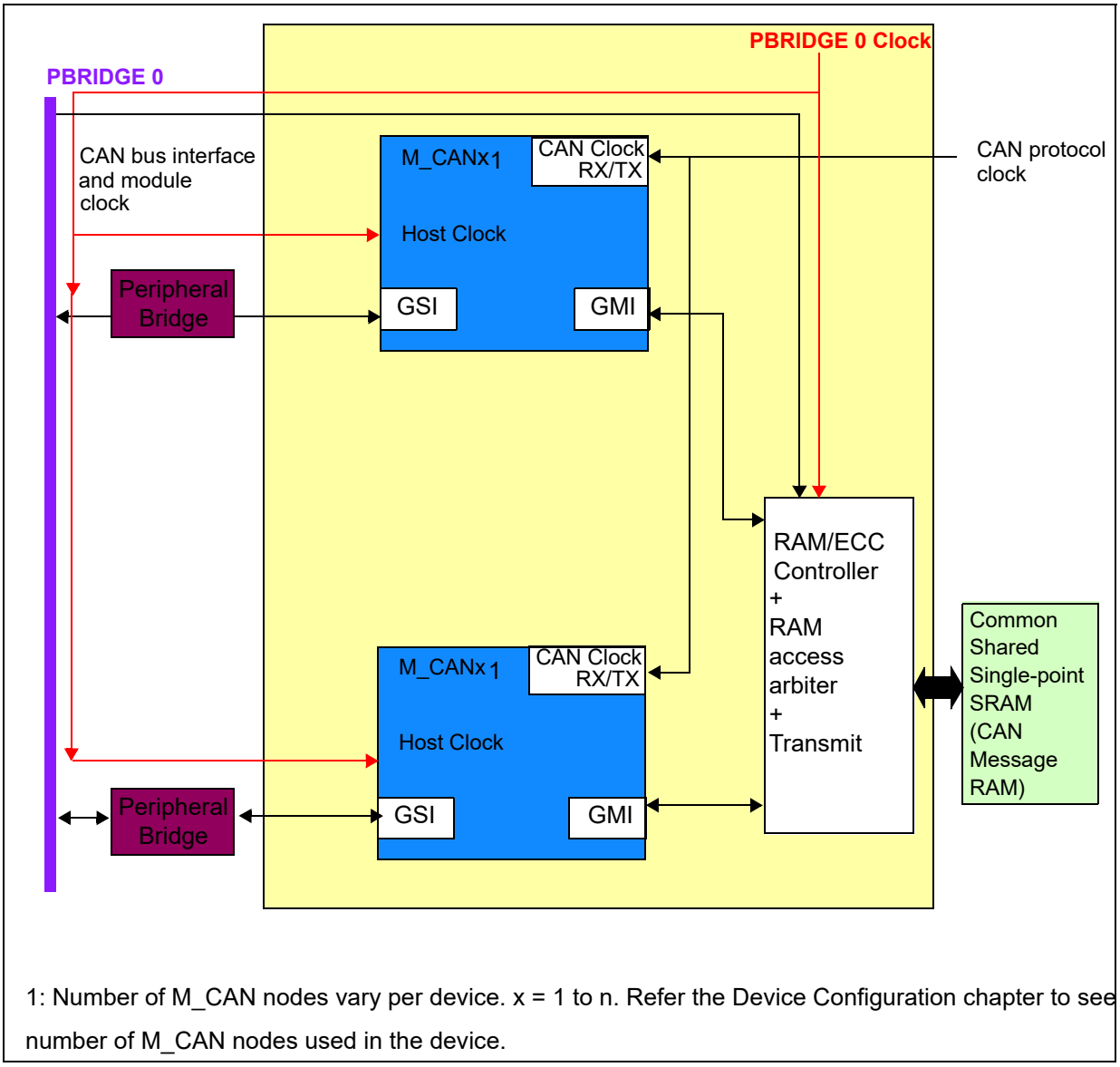
Table 545. References

IP Name	Version
M_CAN IP (1)	Revision 2.0

1. The CAN FD implemented is an early version that is not compatible to the ISO CAN-FD (ISO 11898-1: 2015)

The general CAN subsystem implementation is shown in *Figure 481*. It interfaces with the host processor bus using the peripheral bus. The RAM is not implemented inside the subsystem.

Figure 481. CAN subsystem generic block diagram



1: Number of M_CAN nodes vary per device. x = 1 to n. Refer the Device Configuration chapter to see number of M_CAN nodes used in the device.

44.2 Features

The CAN subsystem consists of the following major blocks:

- Modular CAN cores: The registers of the CAN module can be accessed using the Generic Slave Interface (GSI)
- CAN-RAM arbiter
- SRAM interface and memory organization
- ECC Controller
- Transmit message buffer locking mechanism
  - Trigger memory locking

44.3 Modular CAN cores

CAN functionality conforms to CAN specification V2.0B active for each CAN node.The M_CAN performs communication according to the CAN protocol specification 2.0 part A,B and to CAN FD 1.0.Flexible assignment of Message Objects to nodes.
The bit rate can be programmed to values up to 1 Mbit/s. Additional transceiver hardware is required for connection to the physical layer.

The message storage is intended to be a single-ported Message RAM outside of the module. It is connected to the M_CAN via the Generic Master Interface. Depending on the chosen device, multiple M_CAN controllers can share the same Message RAM.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN core as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_CAN’s clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FMPLL.

44.3.1 Features

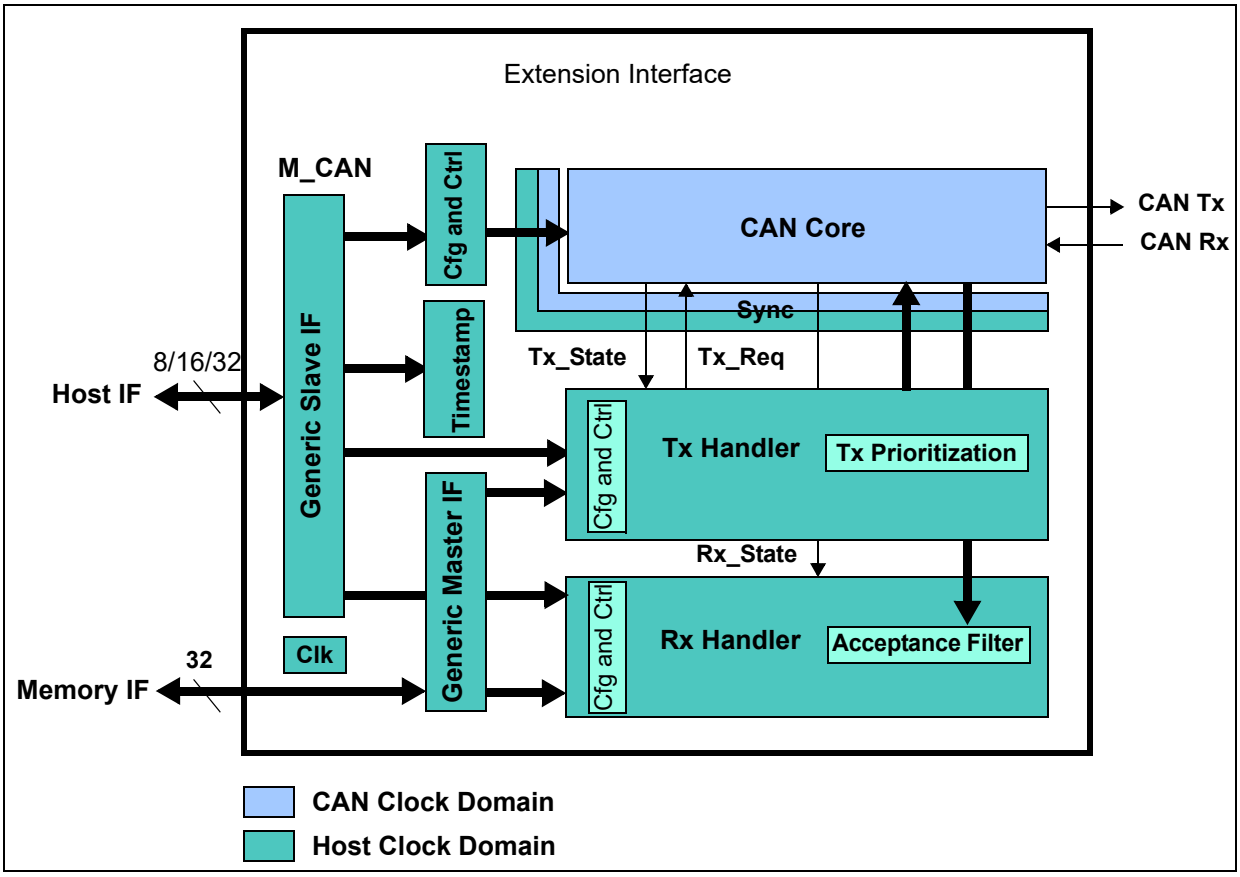
The following are the features of Modular CAN cores.

- Conforms with CAN protocol version 2.0 part A, B and ISO 11898-1
- CAN Flexible data-rate (CAN–FD[n]) protocol with maximum 8 data bytes is supported
- Bit rates up to 1 Mbit/s (in standard CAN mode)
- CAN error logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M_CANs share the same Message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 8-/16-/32-bit Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support
- Debug over M_CAN
- Clock calibration over M_CAN is supported

n. The CAN FD implemented is an early version that is not compatible to the ISO CAN-FD (ISO 11898-1:2015)

44.3.2 Block diagram

Figure 482. M_CAN core block diagram



- CAN core: CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.
- Sync: Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.
- Clk: Synchronizes reset signal to the Host clock domain and to the CAN clock domain.
- Cfg and Ctrl: CAN core related configuration and control bits.
- Interrupt and Timestamp: Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation.
- Tx Handler: Controls the message transfer from the external Message RAM to the CAN core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.
- Rx Handler: Controls the transfer of received messages from the CAN core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored

together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

- Generic Slave Interface: Connects the M_CAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8-/16-/32-bit bus to support a wide range of interconnection structures.
- Generic Master Interface: Connects the M_CAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16 KB × 32 bits. A single M_CAN can use at most 1 KB × 32 bits.
- Extension Interface: All flags from the Interrupt Register IR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M_CAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

44.3.3 Dual clock sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain. Due to the high precision clocking requirements of the CAN core, a separate clock without any modulation has to be provided as CAN clock. The CAN core should be programmed to have at least 8 clocks per bit time, this is for example, 1 Mbaud at CAN clock ≥8 MHz. Even in case of a very high Host clock frequency, the clock frequency of the CAN core must not be higher than 8 MHz.

Within the M_CAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

Note: In order to achieve a stable function of the M_CAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

Caution: Stop this module when the device is in STOP mode. If CAN message reception is mandatory, check that the PLL is unlocked before transmitting any message. Use the external oscillator (XOSC) instead of the internal RC oscillator (IRCOSC) during the STOP mode.

44.3.4 Dual interrupt lines

The module provides two interrupt lines. Interrupts can be routed either to INT 0 or to INT 1. By default all interrupts are routed to interrupt line INT 0. By programming INT 0 and INT 1 bits of the Interrupt Line Enable (ILE) register, the interrupt lines can be enabled or disabled separately.

44.3.5 Memory map and register descriptions

44.3.5.1 Hardware reset description

After hardware reset, the registers of the M_CAN hold the reset values listed in *Table 546*. Additionally the *Bus_Off* state is reset and the M_CAN Tx output is set to *recessive* (HIGH). The value 0x0001 (**bit 0 [INIT]** of the **CCCR register** = ‘1’) in the CC Control Register enables software initialization. The M_CAN does not influence the CAN bus until the CPU resets **bit 0 [INIT]** of the **CCCR register** = ‘0’.

44.3.5.2 Register map

The M_CAN module allocates an address space of 256 bytes. All registers are organized as 32-bit registers. The M_CAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access).

Note: Write access by the Host CPU to registers/bits marked with "P=Protected Write" is possible only when the bit 1 [CCE] and bit 0 [INIT] of the CCCR register is set to "1".

There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

Table 546. M_CAN memory map

Address Offset	Register Name	Access (1)	Reset value	Section/Page
0x0000	CREL—Core Release Register	R	rrrd dddd	*44.3.5.2.1/967*
0x0004	ENDN—Endian Register	R	8765 4321	*44.3.5.2.2/968*
0x0008	Reserved			
0x000C	FBTP—Fast Bit Timing and Prescaler Register	RP	0x000_0A33	*44.3.5.2.3/968*
0x0010	TEST—Test Register	RW	0x0000_0000	*44.3.5.2.4/969*
0x0014	RWD—RAM Watchdog Register	RP	0x0000_0000	*44.3.5.2.5/970*
0x0018	CCCR—CC Control Register	RP	0x0000_0001	*44.3.5.2.6/971*
0x001C	BTP—Bit Timing and Prescaler Register	RP	0x0000_2301	*44.3.5.2.7/973*
0x0020	TSCC—Timestamp Counter Configuration Register	RP	0x0000_0000	*44.3.5.2.8/975*
0x0024	TSCV—Timestamp Counter Value Register	w1c	0x0000_0000	*44.3.5.2.9/976*
0x0028	TOCC—Timeout Counter Configuration Register	RP	0xFFFF_0000	*44.3.5.2.10/976*
0x002C	TOCV—Timeout Counter Value Register	w1c	0x0000_FFFF	*44.3.5.2.11/977*
0x0030–0x003C	Reserved			
0x0040	ECR—Error Counter Register	R	0x0000_0000	*44.3.5.2.12/978*
0x0044	PSR—Protocol Status Register	RS	0x0000_0007	*44.3.5.2.13/979*
0x0048–0x004C	Reserved			
0x0050	IR—Interrupt Register	RW	0x0000_0000	*44.3.5.2.14/981*
0x0054	IE—Interrupt Enable Register	RW	0x0000_0000	*44.3.5.2.15/983*
0x0058	ILS—Interrupt Line Select Register	RW	0x0000_0000	*44.3.5.2.16/986*
0x005C	ILE—Interrupt Line Enable Register	RW	0x0000_0000	*44.3.5.2.17/989*
0x0060–0x007C	Reserved			
0x0080	GFC—Global Filter Configuration Register	RP	0x0000_0000	*44.3.5.2.18/990*
0x0084	SIDFC—Standard ID Filter Configuration Register	RP	0x0000_0000	*44.3.5.2.19/991*

Table 546. M_CAN memory map(Continued)

Address Offset	Register Name	Access (1)	Reset value	Section/Page
0x0088	XIDFC—Extended ID Filter Configuration Register	RP	0x0000_0000	*44.3.5.2.20/992*
0x008C	Reserved			
0x0090	XIDAM—Extended ID AND Mask Register	RP	0x1FFF_FFFF	*44.3.5.2.21/993*
0x0094	HPMS—High Priority Message Status Register	R	0x0000_0000	*44.3.5.2.22/993*
0x0098	NDAT1—New Data 1 Register	RW	0x0000_0000	*44.3.5.2.23/994*
0x009C	NDAT2—New Data 2 Register	RW	0x0000_0000	*44.3.5.2.24/995*
0x00A0	RXF0C—Rx FIFO 0 Configuration Register	RP	0x0000_0000	*44.3.5.2.25/995*
0x00A4	RXF0S—Rx FIFO 0 Status Register	R	0x0000_0000	*44.3.5.2.26/996*
0x00A8	RXF0A—Rx FIFO 0 Acknowledge Register	RW	0x0000_0000	*44.3.5.2.27/997*
0x00AC	RXBC—Rx Buffer Configuration Register	RW	0x0000_0000	*44.3.5.2.28/998*
0x00B0	RXF1C—Rx FIFO 1 Configuration Register	RP	0x0000_0000	*44.3.5.2.29/998*
0x00B4	RXF1S—Rx FIFO 1 Status Register	R	0x0000_0000	*44.3.5.2.30/999*
0x00B8	RXF1A—Rx FIFO 1 Acknowledge Register	RW	0x0000_0000	*44.3.5.2.31/1000*
0x00BC	Reserved			
0x00C0	TXBC—Tx Buffer Configuration Register	RP	0x0000_0000	*44.3.5.2.32/1001*
0x00C4	TXFQS—Tx FIFO/Queue Status Register	R	0x0000_0000	*44.3.5.2.33/1002*
0x00C8	Reserved			
0x00CC	TXBRP—Tx Buffer Request Pending Register	R	0x0000_0000	*44.3.5.2.34/1003*
0x00D0	TXBAR—Tx Buffer Add Request Register	RW	0x0000_0000	*44.3.5.2.35/1004*
0x00D4	TXBCR—Tx Buffer Cancellation Register	RW	0x0000_0000	*44.3.5.2.36/1004*
0x00D8	TXBTO—Tx Buffer Transmission Occurred Register	R	0x0000_0000	*44.3.5.2.37/1005*
0x00DC	TXBCF—Tx Buffer Cancellation Finished Register	R	0x0000_0000	*44.3.5.2.38/1006*
0x00E0	TXBTIE—Tx Buffer Transmission Interrupt Enable	RW	0x0000_0000	*44.3.5.2.39/1006*
0x00E4	TXBCIE—Tx Buffer Cancellation Finished Interrupt Enable Register	RW	0x0000_0000	*44.3.5.2.40/1007*
0x00E8–0x00EC	Reserved			
0x00F0	TXEFC—Tx Event FIFO Configuration Register	RP	0x0000_0000	*44.3.5.2.41/1007*
0x00F4	TXEFS—Tx Event FIFO Status Register	R	0x0000_0000	*44.3.5.2.42/1008*

Table 546. M_CAN memory map(Continued)

Address Offset	Register Name	Access (1)	Reset value	Section/Page
0x00F8	TXEFA—Tx Event FIFO Acknowledge Register	RW	0x0000_0000	*44.3.5.2.43/1009*
0x00FC	Reserved			

1. R = Read, S = Set on read, W = Write, P = Protected Write, C = Clear/preset on write, r = release, d = date

44.3.5.2.1 Core Release Register (CREL)

Address: 0x0000 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset (1)	r	r	r	r	r	r	r	r	r	r	r	r	d	d	d	d

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MON								DAY							
W																
Reset (1)	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d

Figure 483. Core Release Register

1. R = Read; -r = release, -d = time stamp. The coding of revisions depends on the module version used in the device.

Table 547. Core Release Register Field Descriptions

Field	Description
0:3 REL	Core Release One digit, BCD
4:7 STEP	Step of Core Release One digit, BCD
8:11 SUBSTEP	Sub-step of Core Release One digit, BCD
12:15 YEAR	Time Stamp Year One digit, BCD
16:23 MON	Time Stamp Month Two digits, BCD
24:31 DAY	Time Stamp Day Two digits, BCD

44.3.5.2.2 Endian Register (ENDN)

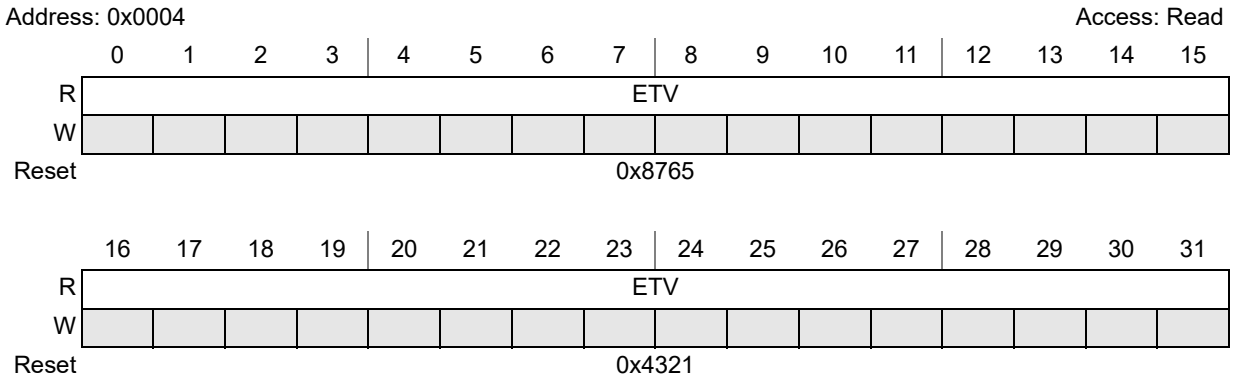


Figure 484. Endian Register

Table 548. Endian Register Field Descriptions

Field	Description
0:31 ETV	Endianness Test Value The endianness test value is 0x87654321.

44.3.5.2.3 Fast Bit Timing and Prescaler Register (FBTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programmed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 1024 M_CAN clock periods. tq = (FBRP + 1) M_CAN clock period.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. **FTSEG2** is Phase_Seg2.

Therefore the length of the bit time is (programmed values) **[FTSEG1 + FTSEG2 + 3]** tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

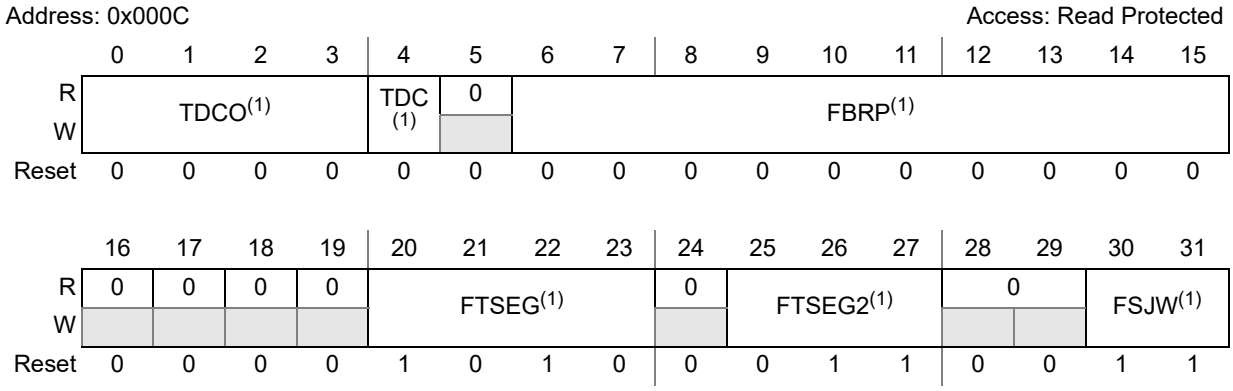


Figure 485. Fast Bit Timing and Prescaler Register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 549. Fast Bit Timing and Prescaler Register

Field	Description
0:3 TDCO	Transceiver Delay Compensation Offset 0x0–0xF— Offset value defining the distance between the measured delay from m_can_tx to m_can_rx and the secondary sample point. Valid values are 0 to 15 M_CAN clock periods.
4 TDC	Transceiver Delay Compensation 0 Transceiver Delay Compensation disabled 1 Transceiver Delay Compensation enabled
6:15 FBRP	Fast Baud Rate Prescaler 0x000–0x3FF— The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
20:23 FTSEG1	Fast time segment before sample point 0x1–0xF— Valid values are 1 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
25:27 FTSEG2	Fast time segment before sample point 0x0–0x7— Valid values are 0 to 7. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
30:31 FSJW	Fast (Re) Synchronization Jump Width 0x0–0x3— Valid values are 0 to 3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a fast bit rate of 500 kbit/s.

44.3.5.2.4 Test Register

Write access to the Test Register has to be enabled by setting CCCR[TEST] to ‘1’. All Test Register functions are set to their reset values when CCCR[TEST] is reset. Loop Back mode and software control of M_CAN Tx pin are hardware test modes. Programming of **TX** ≠ “00” may disturb the message transfer on the CAN bus.

Address: 0x0010 Access: Read-Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TDCV					RX	TX(1)		LBCK(1)	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	—	0	0	0	0	0	0	0

Figure 486. Test Register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 550. Test Register field descriptions

Field	Description
0:14	Reserved
19:23 TDCV	Transceiver Delay Compensation Value 0x00–0x1F— Position of the secondary sample point, defined by the sum of the measured delay from m_can_tx to m_can_rx and FBTP.TDCO. Valid values are 0 to 31 M_CAN core clock periods.
24 RX	Receive Pin Monitors the actual value of pin m_ can_rx 0 The CAN bus is dominant (m_can_rx = ‘0’) 1 The CAN bus is recessive (m_can_rx = ‘1’)
25:26 TX	Control of Transmit Pin 00 Reset value, m_can_tx is controlled by the CAN core, updated at the end of the CAN bit time 01 Sample Point can be monitored at pin m_can_tx 10 Dominant (‘0’) level at pin m_can_tx 11 Recessive (‘1’) at pin m_ can_tx
27 LBCK	Loop Back mode 0 Reset value, Loop Back mode is disabled 1 Loop Back mode is enabled (see *Section 44.3.7.8: Test modes*)
28:31	Reserved

44.3.5.2.5 RAM Watchdog Register (RWD)

The RAM Watchdog monitors the READY output of the Message RAM. A Message RAM access via the M_CAN’s Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by the WDC bits. The counter is reloaded with WDC bits when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag WDI bit of the IR register is set. The RAM Watchdog Counter is clocked by the Host clock .

Address: 0x0014 Access: Read Protected

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDV								WDC(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 487. RAM watchdog register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 551. RAM Watchdog register field descriptions

Field	Description
0:15	Reserved
16:23 WDV	Watchdog Value Actual Message RAM Watchdog Counter Value
24:31 WDC	Watchdog Configuration Start value of the Message RAM Watchdog Counter. With the reset value of “00” the counter is disabled.

44.3.5.2.6 CC Control Register (CCCR)

The following section shows the CAN Control register.For details about setting and resetting of single bits see *Section 44.3.7.1.1: Software initialization*.

Address: 0x0018 Access: Read Protected

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	FACT	LACT	CMR		CME(1)		TEST(1)	DAR(1)	MON(1)	CSR	CSA	ASM(1)	CCE(1)	INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 488. CC Control register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 552. CC Control Register field descriptions

Field	Description
0:17	Reserved
18 FACT	Fast Frame Mode Active 0 Format of transmitted frames defined by LACT 1 Frames transmitted in CAN FD Long format
19 LACT	Long Frame Mode Active 0 Frames transmitted in Standard CAN format 1 Frames transmitted in CAN FD Long format

Table 552. CC Control Register field descriptions(Continued)

Field	Description
Bit 20:21 CMR	CAN Mode Request A change of the CAN operation mode is requested by writing to this bit field. After change to the requested operation mode the bit field is reset to “00” and the status flags FACT and LACT are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CMR is retained until it is overwritten by the next mode change request. Default is normal CAN operation. 00 unchanged 01 Long Frame Mode request 10 Long + Fast Frame Mode request 11 Normal CAN operation request
Bit 22:23 CME	CAN Mode Enable 00 Normal CAN operation according to the ISO11898-1 01 Long Frame Mode enabled 10 Long + Fast Frame Mode enabled 11 Long + Fast Frame Mode enabled **Note:** In case CME = “01” transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CME = “10”/“11” transmission and reception of long and fast CAN FD frames is enabled.
24 TEST	Test Mode Enable 0 Normal operation, register TEST holds reset values 1 Test Mode, write access to register TEST enabled
25 DAR	Disable Automatic Retransmission 0 Automatic retransmission of messages not transmitted successfully enabled 1 Automatic retransmission disabled
26 MON	Bus Monitoring Mode Bit MON can only be set by the Host when both CCE and INIT are set to ‘1’. The bit can be reset by the Host at any time. 0 Bus Monitoring Mode is disabled 1 Bus Monitoring Mode is enabled
27 CSR	Clock Stop Request 0 No clock stop is requested 1 Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN b us reached idle.
28 CSA	Clock Stop Acknowledge 0 No clock stop acknowledged 1 M_CAN may be set in power down by stopping host clock and core clock

Table 552. CC Control Register field descriptions(Continued)

Field	Description
29 ASM	ASM Restricted Operation Mode The Restricted Operation Mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame. In the optional Restricted Operation Mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. If the M_CAN is connected to a Clock Calibration on CAN unit, ASM is controlled by input M_CAN calibration OK (m_can_cok) signal. In case m_can_cok switches to '0', bit ASM is set. When m_can_cok switches back to '1', bit ASM returns to the previously written value. The state of ASM is the written value while input m_can_cok is at '1'. The input is hardwired to '1' when there is no Clock Calibration on CAN unit connected. 0 Normal CAN operation 1 Restricted Operation Mode active
30 CCE	Configuration Change Enable 0 The CPU has no write access to the protected configuration registers 1 The CPU has write access to the protected configuration registers (while CCCR.INIT = '1')
31 INIT	Initialization 0 Normal Operation 1 Initialization is started

Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to ensure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

44.3.5.2.7 Bit Timing and Prescaler Register (BTP)

This register is only writable if bits CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programmed in the range of [4…81] time quanta. The CAN time quantum may be programmed in the range of [1…1024] M_CAN clock periods.

Equation 22 tq = (BRP + 1) M_CAN clock period

TSEG1 is the sum of Prop_Seg and Phase_Seg1. **TSEG2** is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [**TSEG1 + TSEG2 + 3**] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Address: 0x001C Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	BRP(1)									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	TSEG1(1)						TSEG2[3:0](1)				SJW(1)			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

Figure 489. Bit Timing and Prescaler Register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 553. Bit Timing and Prescaler Field Descriptions

Field	Description
0:5	Reserved
6:15 BRP	Baud Rate Prescaler (0x000–0x3FF) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
16:17	Reserved
18:23 TSEG1	Time segment before sample point (0x01–0x3F) Valid values are 1 to 63. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
24:27 TSEG2	The time segment before the sample point (0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
28:31 SJW	(Re) Synchronization Jump Width (0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a bit rate of 500 kbit/s.

Note: The Information Processing Time (IPT) of the M_CAN is 0 tq. Therefore, to conform to the CAN protocol specification, the lengths for phase segment 1 (Phase_Seg1) and phase segment 2 (Phase_Seg2) must be equal.

44.3.5.2.8 Timestamp Counter Configuration Register (TSCC)

Address: 0x0020 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												TCP(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														TSS(1)	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 490. Timestamp Counter Configuration Register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 554. Timestamp Counter Configuration field descriptions

Field	Description
0:11	Reserved
12:15 TCP	Timestamp Counter Prescaler 0x0–0xF Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 to16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. **Note:** With CAN FD an external counter is required for timestamp generation (TSS = “10”)
16:29	Reserved
30:31 TSS	Timestamp Select 00 Timestamp counter value always 0x0000 01 Timestamp counter value incremented according to **TCP** 10 Reserved 11 Same as “00”

44.3.5.2.9 Timestamp Counter Value Register (TSCV)

Address: 0x0024 Access: w1c

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TSC															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 491. Timestamp counter value register

Note: A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.

44.3.5.2.10 Timeout Counter Configuration Register (TOCC)

For a description of the Timeout Counter see *Section 44.3.9: Timeout counter*.

Address: 0x0028 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TOP(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													TOS(1)		ETOC (1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 492. Timeout counter configuration register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 555. Timeout Counter Configuration Field Descriptions

Field	Description
0:15 TOP[15:0]	Timeout Period Start value of the Timeout Counter (down-counter). Configures the Timeout Period.
16:28	Reserved

Table 555. Timeout Counter Configuration Field Descriptions(Continued)

Field	Description
29:30 TOS	Timeout Select When operating in Continuous mode, a write to **TOCV** presets the counter to the value configured by **TOCC[TOP]** and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by **TOCC[TOP]**. Down-counting is started when the first FIFO element is stored. 00 Continuous operation 01 Timeout controlled by Tx Event FIFO 10 Timeout controlled by Rx FIFO 0 11 Timeout controlled by Rx FIFO 1
31 ETOC	Enable Timeout Counter 0 Timeout Counter disabled 1 Timeout Counter enabled **Note:** For use of timeout function with CAN FD see *Section 44.3.9: Timeout counter*.

44.3.5.2.11 Timeout Counter Value Register (TOCV)

Address: 0x002C Access: w1c

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TOC															
W	w1c															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 493. Timeout counter value register

Table 556. Timeout counter value field descriptions

Field	Description
0:15	Reserved
16:31 TOC	The Timeout Counter is decremented in multiples of CAN bit times [1 to 16] depending on the configuration of TSCC.TCP. When decremented to zero, interrupt flag IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS.

44.3.5.2.12 Error Counter Register (ECR)

Address: 0x0040 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CEL(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RP	REC							TEC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 494. Error counter register

1. Access type is RX: reset on read.

Table 557. Error counter register field descriptions

Field	Description
0:7	Reserved
8:15 CEL	CAN Error Logging The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag ELO bit of the IR register.
16 RP	Receive Error Passive 0 The Receive Error Counter is below the error passive level of 128 1 The Receive Error Counter has reached the error passive level of 128
17:23 REC	Receive Error Counter Actual state of the Receive Error Counter, values between 0 and 127
24:31 TEC	Transmit Error Counter Actual state of the Transmit Error Counter, values between 0 and 255

Note: When CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

44.3.5.2.13 Protocol Status Register (PSR)

Address: 0x0044 Access: RS

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	RED L(1)	RBR S(1)	RESI (1)	FLEC(2)			BO	EW	EP	ACT		LEC(2)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 495. Protocol status register (PSR)

1. Access type is RX: reset on read.
2. Access type is RS: set on read.

Table 558. PSR descriptions

Field	Description
0:17	Reserved
18 REDL	Received CAN FD Message with EDL flag 0 Since this bit was reset by the CPU, no CAN FD message has been received 1 Message in CAN FD format with EDL flag set has been received
19 RBRS	BRS flag of last received CAN FD Message This bit is set together with REDL, independent of acceptance filtering. 0 Last received CAN FD message did not have its BRS flag set. 1 Last received CAN FD message had its BRS flag set.
20 RESI	ESI CAN FD Message with ESI flag This bit is set together with REDL, independent of acceptance filtering. 0 Last received CAN FD message did not have its ESI flag set. 1 Last received CAN FD message had its ESI flag set
21:23 FLEC	Fast Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error.
24 BO	Bus_Off Status 0 The M_CAN is not Bus_Off 1 The M_CAN is in Bus_Off state
25 EW	Warning Status 0 Both error counters are below the Error_Warning limit of 96 1 At least one of error counter has reached the Error_Warning limit of 96

Table 558. PSR descriptions(Continued)

Field	Description
26 EP	Error Passive 0 The M_CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1 The M_CAN is in the Error_Passive state
27:28 ACT	Activity Monitors the module's CAN communication state. 00 Synchronizing – node is synchronizing on CAN communication 01 Idle—node is neither receiver nor transmitter 10 Receiver—node is operating as receiver 11 Transmitter—node is operating as transmitter
29:31 LEC	Last Error Code The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. 000 No Error: No error occurred since LEC has been reset by successful reception or transmission. 001 Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010 Form Error: A fixed format part of a received frame has the wrong format. 011 AckError: The message transmitted by the M_CAN was not acknowledged by another node. 100 Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant. 101 Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

Note: When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error

Note: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT.] If the device goes Bus_Off, it will set CCCR[INIT] of its own accord, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit 0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences.

44.3.5.2.14 Interrupt Register (IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. The configuration of **IE** controls whether an interrupt is generated. The configuration of **ILS** controls on which interrupt line an interrupt is signaled.

Address: 0x0050 Access: w1c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STE	FOE	ACKE	BE	CRCE	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	UMD	TSW
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	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 496. Interrupt register

Table 559. Interrupt register field descriptions

Field	Description
0 STE	Stuff Error 0 No Stuff Error detected 1 More than 5 equal bits in a sequence occurred
1 FOE	Format Error 0 No Format Error detected 1 A fixed format part of a received frame has the wrong format
2 ACKE	Acknowledge Error 0 No Acknowledge Error detected 1 A transmitted message was not acknowledged by another node
3 BE	Bit Error 0 No Bit Error detected 1 Device wanted to send a rec / dom level, but monitored bus level was dom / rec
4 CRCE	CRC Error 0 No CRC Error detected 1 Received CRC did not match the calculated CRC
5 WDI	Watchdog Interrupt 0 No Message RAM Watchdog event occurred 1 Message RAM Watchdog event due to missing READY
6 BO	Bus_Off Status 0 Bus_Off status unchanged 1 Bus_Off status changed

Table 559. Interrupt register field descriptions(Continued)

Field	Description
7 EW	Warning Status 0 Error_Warning status unchanged 1 Error_Warning status changed
8 EP	Error Passive 0 Error_Passive status unchanged 1 Error_Passive status changed
9 ELO	Error Logging Overflow 0 CAN Error Logging Counter did not overflow 1 Overflow of CAN Error Logging Counter occurred
10 BEU	Bit Error Uncorrected Message RAM bit error detected, uncorrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets CCCR[INIT] to '1'. This is done to avoid transmission of corrupted data. 0 No bit error detected when reading from Message RAM 1 Bit error detected, uncorrected (e.g. parity logic)
11 BEC	Bit Error Corrected Message RAM bit error detected and corrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected (e.g. ECC)
12 DRX	Message stored to Dedicated Rx Buffer The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 No Rx Buffer updated 1 At least one received message stored into a Rx Buffer
13 TOO	Timeout Occurred 0 No timeout 1 Timeout reached
14 UMD	Unprocessed Message Discarded When a new message is received while the acceptance filtering process for the previously received message has not yet completed, this message is discarded. 0 No unprocessed message discarded 1 Unprocessed message discarded
15 TSW	Timestamp Wraparound 0 No timestamp counter wrap-around 1 Timestamp counter wrapped around
16 TEFL	Tx Event FIFO Event Lost 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
17 TEFF	Tx Event FIFO Full
18 TEFW	Tx Event FIFO Watermark Reached 0 Tx Event FIFO fill level below watermark 1 Tx Event FIFO fill level reached watermark

Table 559. Interrupt register field descriptions(Continued)

Field	Description
19 TEFN	Tx Event FIDO New Entry 0 Tx Event FIFO unchanged 1 Tx Handler wrote Tx Event FIFO element
20 TFE	Tx FIFO Empty 0 Tx FIFO non-empty 1 Tx FIFO empty
21 TCF	Transmission Cancellation Finished 0 No transmission cancellation finished 1 Transmission cancellation finished
22 TC	Transmission Completed 0 No transmission completed 1 Transmission completed
23 HPM	High Priority Message 0 No high priority message received 1 High priority message received
24 RF1L	Rx FIFO 1 Message Lost 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
25 RF1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
26 RF1W	Rx FIFO 1 Watermark Reached 0 Rx FIFO 1 fill level below watermark 1 Rx FIFO 1 fill level reached watermark
27 RF1N	Rx FIFO 1 New Message 0 No new message written to Rx FIFO 1 1 New message written to Rx FIFO 1
28 RF0L	Rx FIFO 0 Message Lost 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
29 RF0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
30 RF0W	Rx FIFO 0 Watermark Reached 0 Rx FIFO 0 fill level below watermark 1 Rx FIFO 0 fill level reached watermark
31 RF0N	Rx FIFO 0 New Message 0 No new message written to Rx FIFO 0 1 New message written to Rx FIFO 0

44.3.5.2.15 Interrupt Enable Register (IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt register will be signaled on an interrupt line.

Address: 0x0054 Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	STEE	FOEE	ACKEE	BEE	CRCEE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	UMDE	TSWE
Reset	0	0	0	0	0	0	0	0	0	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	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 497. Interrupt enable register

Table 560. Interrupt enable register field descriptions

Field	Description
0 STEE	Stuff Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
1 FOEE	Format Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
2 ACKEE	Acknowledge Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
3 BEE	Bit Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
4 CRCEE	CRC Error Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
5 WDIE	Watchdog Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
6 BOE	Bus_Off Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
7 EWE	Warning Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
8 EPE	Error Passive Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 560. Interrupt enable register field descriptions(Continued)

Field	Description
9 ELOE	Error Logging Overflow Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
10 BEUE	Bit Error Uncorrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
11 BECE	Bit Error Corrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
12 DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable
13 TOOE	Timeout Occurred Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
14 UMDE	Unprocessed Message Discarded Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
15 TSWE	Timestamp Wraparound Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
16 TEFLE	Tx Event FIFO Event Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
17 TEFFE	Tx Event FIFO Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
18 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
19 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
20 TFEE	Tx FIFO Empty Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
21 TCFE	Transmission Cancellation Finished Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
22 TCE	Transmission Completed Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 560. Interrupt enable register field descriptions(Continued)

Field	Description
23 HPME	High Priority Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
24 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
25 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
26 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
27 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
28 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
29 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
30 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
31 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

44.3.5.2.16 Interrupt Line Select Register (ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines.

Address: 0x0058 Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	STEL	FOEL	ACKEL	BEL	CRCEL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	UMDL	TSWL
Reset	0	0	0	0	0	0	0	0	0	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	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 498. Interrupt line select register

Table 561. Interrupt line select register field descriptions

Field	Description
0 STEL	Stuff Error Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
1 FOEL	Format Error Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
2 ACKEL	Acknowledge Error Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
3 BEL	Bit Error Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
4 CRCEL	CRC Error Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
5 WDIL	Watchdog Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
6 BOL	Bus_Off Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
7 EWL	Warning Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
8 EPL	Error Passive Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 561. Interrupt line select register field descriptions(Continued)

Field	Description
9 ELOL	Error Logging Overflow Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
10 BEUL	Bit Error Uncorrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
11 BECL	Bit Error Corrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
12 DRXL	Message stored to Dedicated Rx Buffer Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
13 TOOL	Timeout Occurred Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
12 UMDL	UMDL: Unprocessed Message Discarded Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
15 TSWL	TSWL: Timestamp Wraparound Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
16 TEFLL	Tx Event FIFO Event Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
17 TEFFL	Tx Event FIFO Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
18 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
19 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
20 TFEL	Tx FIFO Empty Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
21 TCFL	Transmission Cancellation Finished Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
22 TCL	Transmission Completed Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 561. Interrupt line select register field descriptions(Continued)

Field	Description
23 HPML	High Priority Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
24 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
25 RF1FL	Rx FIFO 1 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
26 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
27 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
28 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
29 RF0FL	Rx FIFO 0 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
30 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
31 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

44.3.5.2.17 Interrupt Line Enable Register (ILE)

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1.

Address: 0x005C Access: RW

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EINT1	EINT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 499. Interrupt line enable register

Table 562. Interrupt line enable register field descriptions

Field	Description
0:29 STEL	Reserved
30 EINT1	Enable Interrupt Line 1 0 Interrupt line disabled 1 Interrupt line enabled
31 EINT0	Enable Interrupt Line 0 0 Interrupt line disabled 1 Interrupt line enabled

44.3.5.2.18 Global Filter Configuration Register (GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in *Figure 536* and *Figure 537*.

Address: 0x0080 Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ANFS(1)		ANFE (1)		RRFS (1)	RRFE (1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 500. Global filter configuration register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 563. Global filter configuration register field descriptions

Field	Description
0:25	Reserved
26:27 ANFS	Accept Non-matching Frames Standard Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject

Table 563. Global filter configuration register field descriptions(Continued)

Field	Description
28:29 ANFE	ANFE[1:0]: Accept Non-matching Frames Extended Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
30 RRFS	Reject Remote Frames Standard 0 Filter remote frames with 11-bit standard IDs 1 Reject all remote frames with 11-bit standard IDs
31 RRFE	Reject Remote Frames Extended 0 Filter remote frames with 29-bit extended IDs 1 Reject all remote frames with 29-bit extended IDs

44.3.5.2.19 Standard ID Filter Configuration Register (SIDFC)

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in Figure 530.

Address: 0x0084 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	LSS(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLSSA[15:2](1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 501. Standard ID Filter Configuration register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 564. Standard ID Filter Configuration register field descriptions

Field	Description
0:7 STEL	Reserved
8:15 LSS	List Size Standard 0 No standard Message ID filter 1–128 Number of standard Message ID filter elements >128 Values greater than 128 are interpreted as 128

Table 564. Standard ID Filter Configuration register field descriptions(Continued)

Field	Description
16:29 FLSSA	Filter List Standard Start Address Start address of standard Message ID filter list (32-bit word address, see *Figure 526*).
30:31	Reserved

44.3.5.2.20 Extended ID Filter Configuration Register (XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in Figure 531.

Address: 0x0088 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LSE(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLESA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 502. Extended ID Filter Configuration register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 565. Extended ID Filter Configuration register field descriptions

Field	Description
0:8	Reserved
9:15 LSE	List Size Extended 0 No extended Message ID filter 1–64 Number of extended Message ID filter elements >64 Values greater than 64 are interpreted as 64
16:29 FLESA	Filter List Extended Start Address Start address of extended Message ID filter list (32-bit word address, see *Figure 526*).
30:31	Reserved

44.3.5.2.21 Extended ID and Mask Register (XIDAM)

Address: 0x0090 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	EIDM(1)												
W																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIDM(1)															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 503. Extended ID and Mask register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 566. Extended ID and Mask register field descriptions

Field	Description
0:2	Reserved
3:31 EIDM	Extended ID Mask For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.

44.3.5.2.22 High Priority Message Status Register (HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address: 0x0094 Access: R

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLST	FIDX							MSI		BIDX					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 504. High Priority Message Status register

Table 567. High Priority Message Status register field descriptions

Field	Description
0:15	Reserved
16 FLST	Filter List Indicates the filter list of the matching filter element. 0 Standard Filter List 1 Extended Filter List
17:23 FIDX	Filter Index Index of matching filter element. Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
24:25 MSI	Message Storage Indicator 00 No FIFO selected 01 FIFO overrun 10 Message stored in FIFO 0 11 Message stored in FIFO 1
26:31 BIDX	Buffer Index Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = '1'.

44.3.5.2.23 New Data 1 Register (NDAT1)

Address: 0x0098 Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
Reset	0	0	0	0	0	0	0	0	0	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	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 505. NDAT1 register

Table 568. NDAT1 register field descriptions

Field	Description
0 ND	New Data[31:0] The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

44.3.5.2.24 New Data 2 Register (NDAT2)

Address: 0x009C Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
Reset	0	0	0	0	0	0	0	0	0	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	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32	ND31
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 506. NDAT2 register

Table 569. NDAT2 register field descriptions

Field	Description
0 ND	New Data[63:32] The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

44.3.5.2.25 Rx FIFO 0 Configuration Register (RXF0C)

Address: 0x00A0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F0WM(1)							0	F0S(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F0SA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 507. Rx FIFO 0 Configuration register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 570. Rx FIFO 0 Configuration register field descriptions

Field	Description
0	Reserved
1:7 F0WM	Rx FIFO 0 Watermark 0 Watermark interrupt disabled 1-64 Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) >64 Watermark interrupt disabled
8:9	Reserved
10:15 F0S	Rx FIFO 0 Size 0 No Rx FIFO 0 1-64 Number of Rx FIFO 0 elements >64 Values greater than 64 are interpreted as 64 The Rx FIFO 0 elements are indexed from 0 to F0S-1
16:29 F0SA	Rx FIFO 0 Start Address Start address of Rx FIFO 0 in Message RAM (32-bit word address, *Figure 526*).
30:31	Reserved

44.3.5.2.26 Rx FIFO 0 Status Register (RXF0S)

Address: 0x00A4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	RF0L	F0F	0	0	F0PI					
W																
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F0GI						0	F0FL						
W																
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 508. Rx FIFO 0 Status Register

Table 571. Rx FIFO 0 Status register field descriptions

Field	Description
0:5	Reserved
6 RF0L	Rx FIFO 0 Message Lost This bit is a copy of interrupt flag **IR[RF0L]**. When **IR[RF0L]** is reset, this bit is also reset. 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
7 F0F	Rx FIFO 0 Full 0Rx FIFO 0 not full 1Rx FIFO 0 full
8:9	Reserved

Table 571. Rx FIFO 0 Status register field descriptions(Continued)

Field	Description
10:15 F0PI	Rx FIFO 0 Put Index Rx FIFO 0 write index pointer, range 0 to 63.
16:17	Reserved
18:23 F0GI	Rx FIFO 0 Get Index Rx FIFO 0 read index pointer, range 0 to 63.
24	Reserved
25:31 F0FL	Rx FIFO 0 Fill Level Number of elements stored in Rx FIFO 0, range 0 to 64.

44.3.5.2.27 Rx FIFO 0 Acknowledge Register (RXF0A)

Address: 0x00A8 Access: RW

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F0AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 509. Rx FIFO 0 Acknowledge register

Table 572. Rx FIFO 0 Acknowledge register field descriptions

Field	Description
0:25	Reserved
26:31 F0AI	Rx FIFO 0 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to **F0AI**. This will set the Rx FIFO 0 Get Index **RXF0S[F0GI]** to **F0AI** + 1 and update the FIFO 0 Fill Level **RXF0S[F0FL]**.

44.3.5.2.28 Rx Buffer Configuration Register (RXBC)

Address: 0x00AC Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RBSA														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 510. Rx FIFO 0 Configuration register

Table 573. Rx FIFO 0 Configuration register field descriptions

Field	Description
0:15	Reserved
16:29 RBSA	Rx Buffer Start Address Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C.
30:31	Reserved

44.3.5.2.29 Rx FIFO 1 Configuration Register (RXF1C)

Address: 0x00B0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F1WM(1)							0	F1S						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F1SA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 511. Rx FIFO 0 Configuration register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 574. Rx FIFO 0 Configuration register field descriptions

Field	Description
0	Reserved
1:7 F1WM	Rx FIFO 1 Watermark 0 Watermark interrupt disabled 1-64 Level for Rx FIFO 1 watermark interrupt **IR[RF1W]** >64 Watermark interrupt disabled
8	Reserved
9:15 F1S	F1S[6:0]: Rx FIFO 1 Size 0 No Rx FIFO 1 1-64 Number of Rx FIFO 1 elements >64 Values greater than 64 are interpreted as 64 The Rx FIFO 1 elements are indexed from 0 to **F1S** - 1
16:29 F1SA	Rx FIFO 1 Start Address Start address of Rx FIFO 1 in Message RAM (32-bit word address, see Figure 526).
30:31	Reserved

44.3.5.2.30 Rx FIFO 1 Status Register (RXF1S)

Address: 0x00B4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DMS		0	0	0	0	RF1L	F1F	0	0	F1PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F1GI						0	F1FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 512. Rx FIFO 1 Status register

Table 575. Rx FIFO 1 Status register field descriptions

Field	Description
0:1 DMS	Debug Message Status 00 Idle state, wait for reception of debug messages, DMA request is cleared 01 Debug message A received 10 Debug messages A, B received 11 Debug messages A, B, C received, DMA request is set
2:5	Reserved

Table 575. Rx FIFO 1 Status register field descriptions(Continued)

Field	Description
6 RF1L	Rx FIFO 1 Message Lost This bit is a copy of interrupt flag **IR[RF1L]**. When **IR[RF1L]** is reset, this bit is also reset. 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
7 F1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
8:9	Reserved
10:15 F1P1	Rx FIFO 1 Put Index Rx FIFO 1 write index pointer, range 0 to 63.
16:17	Reserved
18:23 F1G1	Rx FIFO 1 Get Index Rx FIFO 1 read index pointer, range 0 to 63.
24	Reserved
25:31 F1FL	Rx FIFO 1 Fill Level Number of elements stored in Rx FIFO 1, range 0 to 64.

44.3.5.2.31 Rx FIFO 1 Acknowledge Register (RXF1A)

Address: 0x00B8 Access: RW

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F1AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 513. Rx FIFO 1 Acknowledge register

Table 576. Rx FIFO 1 Acknowledge register field descriptions

Field	Description
0:25	Reserved
26:31 F1AI	Rx FIFO 1 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to **F1AI**. This will set the Rx FIFO 1 Get Index **RXF1S[F1GI]** to **F1AI** + 1 and update the FIFO 1 Fill Level **RXF1S[F1FL]**.

44.3.5.2.32 Tx Buffer Configuration Register (TXBC)

Address: 0x00C0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TFQM(1)	TFQS(1)						0	0	NDTB(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 514. Tx Buffer Configuration register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 577. Tx Buffer Configuration register field descriptions

Field	Description
0	Reserved
1 TQFM	Tx FIFO/Queue Mode 0 Tx FIFO operation 1 Tx Queue operation
2:7 TQFS	Tx FIFO/Queue Size 0 No Tx FIFO/Queue 1-32 Number of Tx Buffers used for Tx FIFO/Queue >32 Values greater than 32 are interpreted as 32
8:9	Reserved
10:15 NDTB	Number of Dedicated Transmit Buffers 0 No Dedicated Tx Buffers 1-32 Number of Dedicated Tx Buffers >32 Values greater than 32 are interpreted as 32
16:29 TBSA	Tx Buffers Start Address Start address of Tx Buffers section in Message RAM (32-bit word address, see Figure 526).
30:31	Reserved

Note: Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

44.3.5.2.33 Tx FIFO/Queue Status Register (TXFQS)

Address: 0x00C4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	TFQF	TFQPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TFGI					0	0	TFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 515. Tx FIFO/Queue Status register

Table 578. Tx Buffer Configuration register field descriptions

Field	Description
0:9	Reserved
10 TFQF	Tx FIFO/Queue Full 0 Tx FIFO/Queue not full 1 Tx FIFO/Queue full
11:15 TFQPI	Tx FIFO/Queue Put Index Tx FIFO/Queue write index pointer, range 0 to 31.
16:18	Reserved
19:23 TFGI	Tx FIFO Get Index Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (**TXBC[TFQM]** = ‘1’).
24:25	Reserved
26:31 TFFL	Tx FIFO Free Level Number of consecutive free Tx FIFO elements starting from **TFGI**, range 0 to 32. Read as zero when Tx Queue operation is configured (**TXBC[TFQM]** = ‘1’)

Note: In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers. Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

44.3.5.2.34 Tx Buffer Request Pending Register (TXBRP)

Address: 0x00CC Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 516. Tx Buffer Request Pending register

Table 579. Tx Buffer Request Pending register field descriptions

Field	Description
0:31 TRP	Transmission Request Pending Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register **TXBAR**. The bits are reset after a requested transmission has completed or has been canceled via register **TXBCR**. **TXBRP** bits are set only for those Tx Buffers configured via **TXBC**. After a **TXBRP** bit has been set, a Tx scan (see *Section 44.3.13.7: Tx Event handling*) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register **TXBRP**. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding **TXBRP** bit has been reset. After a cancellation has been requested, a finished cancellation is signaled via **TXBCF** – after successful transmission together with the corresponding **TXBTO** bit – when the transmission has not yet been started at the point of cancellation – when the transmission has been aborted due to lost arbitration – when an error occurred during frame transmission In DAR mode all transmissions are automatically canceled if they are not successful. The corresponding **TXBCF** bit is set for all unsuccessful transmissions. 0 No transmission request pending 1 Transmission request pending

Note: TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is canceled immediately, the corresponding TXBRP bit is reset.

44.3.5.2.35 Tx Buffer Add Request Register (TXBAR)

Address: 0x00D0 Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 517. Tx Buffer Add Request register

Table 580. Tx Buffer Add Request register field descriptions

Field	Description
0:31 AR	Add Request Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to **TXBAR**. **TXBAR** bits are set only for those Tx Buffers configured via **TXBC**. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed. 0 No transmission request added 1 Transmission requested added

Note: If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

44.3.5.2.36 Tx Buffer Cancellation Request Register (TXBCR)

Address: 0x00D4 Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 518. Tx Buffer Cancellation Request register (TXBCR)

Table 581. TXBCR field descriptions

Field	Description
0:31 CR	Cancellation Request Each Tx Buffer has its own Cancellation Request bit. Writing a ‘1’ will set the corresponding Cancellation Request bit; writing a ‘0’ has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset. 0 No cancellation pending 1 Cancellation pending

44.3.5.2.37 Tx Buffer Transmission Occurred Register (TXBTO)

Address: 0x00D8 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 519. Tx Buffer Transmission Occurred register

Table 582. Tx Buffer Transmission Occurred field description

Field	Description
[0:31] TO	Transmission Occurred. Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding **TXBRP** bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a ‘1’ to the corresponding bit of register **TXBAR**. 0 No transmission occurred 1 Transmission occurred

44.3.5.2.38 Tx Buffer Cancellation Finished Register (TXBCF)

Address: 0x00DC Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CF															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CF															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 520. Tx Buffer Cancellation Finished register

Table 583. Tx Buffer Cancellation Finished field description

Field	Description
[0:31] CF	Cancellation Finished. Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding **TXBRP** bit is cleared after a cancellation was requested via TXBCR. In case the corresponding **TXBRP** bit was not set at the point of cancellation, **CF** is set immediately. The bits are reset when a new transmission is requested by writing a ‘1’ to the corresponding bit of register **TXBAR**. 0 No transmit buffer cancellation 1 Transmit buffer cancellation finished

44.3.5.2.39 Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Address: 0x00E0 Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TIE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TIE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 521. Tx Buffer Transmission Interrupt Enable register

Table 584. Tx Buffer Transmission Interrupt field description

Field	Description
[0:31] TIE	Transmission Interrupt Enable. Each Tx Buffer has its own Transmission Interrupt Enable bit. 0 Transmission interrupt disabled 1 Transmission interrupt enable

44.3.5.2.40 Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Address: 0x00E4 Access: RW

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CFIE															
Reset	0	0	0	0	0	0	0	0	0	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	CFIE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 522. Tx Buffer Cancellation Finished Interrupt Enable register

Table 585. Tx Buffer Cancellation Finished Interrupt Enable field description

Field	Description
[0:31] CFIE	Cancellation Finished Interrupt Enable. Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 Cancellation finished interrupt disabled 1 Cancellation finished interrupt enabled

44.3.5.2.41 Tx Event FIFO Configuration Register (TXEFC)

Address: 0x00F0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		EFWM(1)						0		EFS(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EFSA(1)														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 523. Tx Event FIFO Configuration register

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to “1”.

Table 586. Tx Event FIFO Configuration field description

Field	Description
[0:1]	Reserved.
[2:7] EFWM	Event FIFO Watermark. 0 Watermark interrupt disabled 1–32 Level for Tx Event FIFO watermark interrupt (IR[TEFW]) >32 Watermark interrupt disabled
[8:9]	Reserved.
[10:15] EFS	Event FIFO Size. 0 Tx Event FIFO disabled 1–32 Number of Tx Event FIFO elements >32 Values greater than 32 are interpreted as 32 The Tx Event FIFO elements are indexed from 0 to EFS - 1
[16:29] EFSA	Event FIFO Start Address. Start address of Tx Event FIFO in Message RAM (32-bit word address,*Figure 526*.)
[30:31]	Reserved.

44.3.5.2.42 Tx Event FIFO Status Register (TXEFS)

Address: 0x00F4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						TEFL	EFF	0			EFPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			EFGI					0		EFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 524. Tx Event FIFO Status register

Table 587. Tx Event FIFO Status field description

Field	Description
[0:5]	Reserved.
6 TEFL	Tx Event FIFO Element Lost. This bit is a copy of interrupt flag **IR[TEFL]**. When **IR[TEFL]** is reset, this bit is also reset. 0No Tx Event FIFO element lost 1Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
7 EFF	Event FIFO Full. 0Tx Event FIFO not full 1Tx Event FIFO full
[8:10]	Reserved.

Table 587. Tx Event FIFO Status field description(Continued)

Field	Description
[11:15] EFPI	Event FIFO Put Index. Tx Event FIFO write index pointer, range 0 to 31.
[16:18]	Reserved
[19:23] EFGI	Event FIFO Get Index. Tx Event FIFO read index pointer, range 0 to 31.
[24:25]	Reserved.
[26:31] EFFL	Event FIFO Fill Level. Number of elements stored in Tx Event FIFO, range 0 to 32.

44.3.5.2.43 Tx Event FIFO Acknowledge Register (TXEFA)

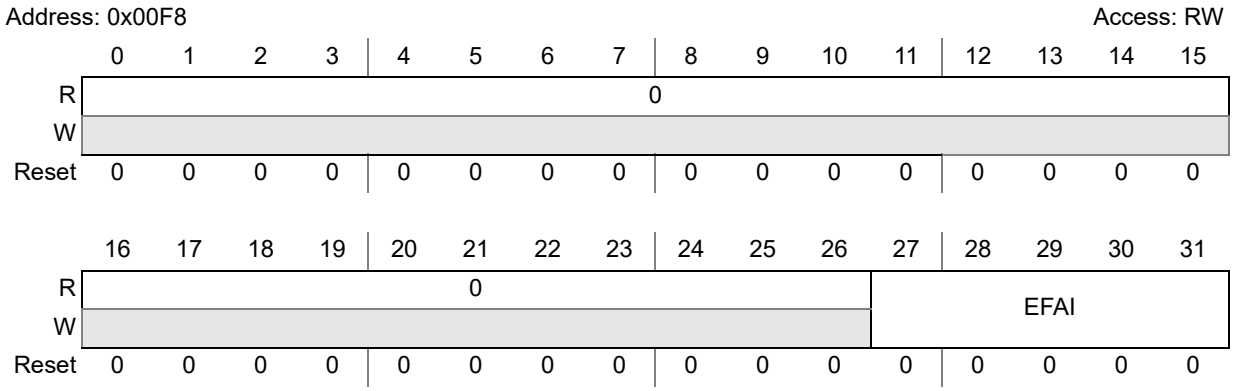


Figure 525. Tx Event FIFO Acknowledge register

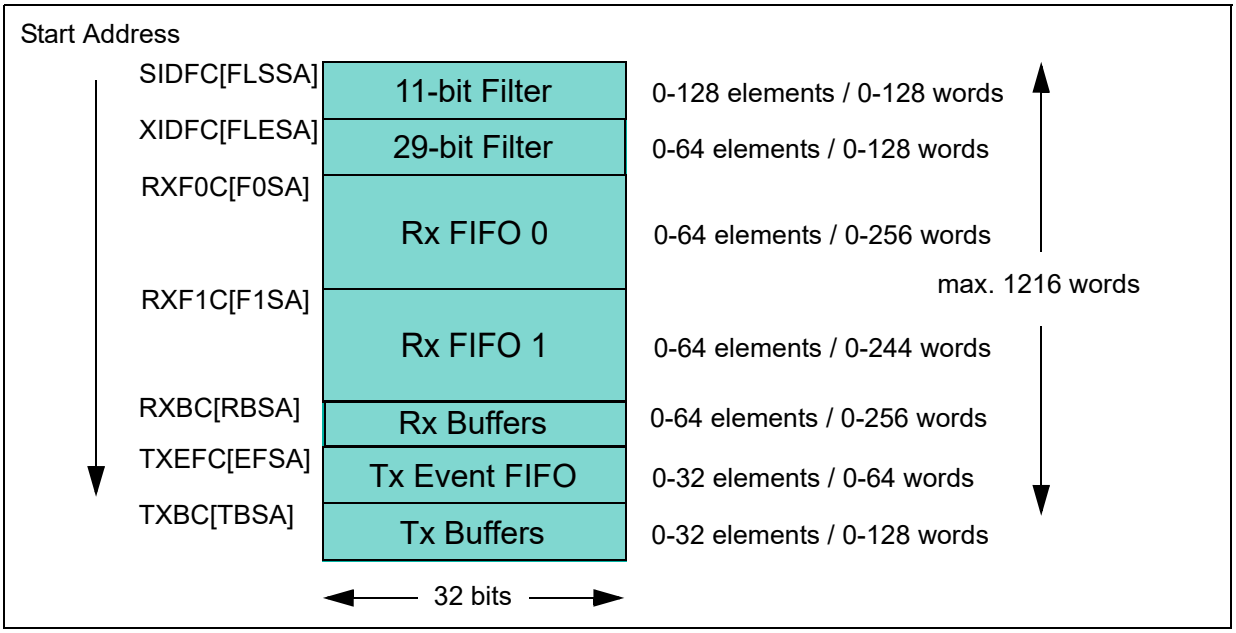
Table 588. Tx Event FIFO Acknowledge field description

Field	Description
[0:26]	Reserved.
[27:31] EFAI	Event FIFO Acknowledge Index. After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index **TXEFS[EFGI]** to EFAI + 1 and update the FIFO 0 Fill Level **TXEFS[EFFL]**.

44.3.6 Message RAM

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_CAN module can be configured to allocate up to 1216 words in the Message RAM. It is not necessary to configure each of the sections listed in the following figure, nor is there any restriction with respect to the sequence of the sections.

Figure 526. Message RAM Configuration



When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configured start addresses are 32-bit word addresses.

Note: The M_CAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

44.3.6.1 Rx Buffer and FIFO element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in the following figure.

Figure 527. Rx FIFO Element

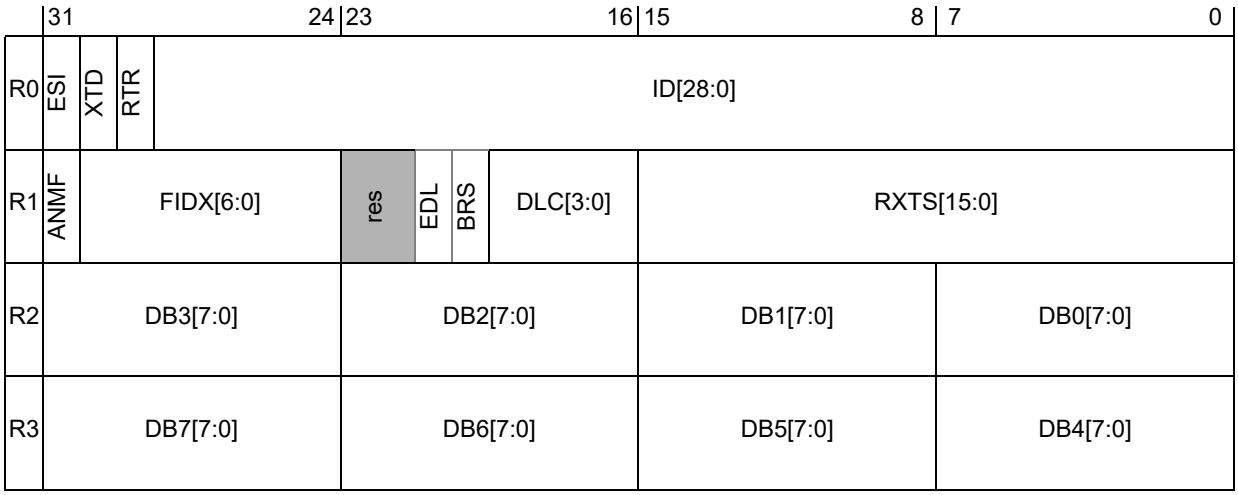


Table 589. Rx FIFO Element descriptions

R0 Bit 31 ESI: Error State Indicator	0 Transmitting node is error active 1 Transmitting node is error passive
R0 Bit 30 XTD: Extended Identifier	Signals to the Host whether the received frame has a standard or extended identifier. 0 11-bit standard identifier 1 29-bit extended identifier
R0 Bit 29 RTR: Remote Transmission Request	Signals to the Host whether the received frame is a data frame or a remote frame. 0 Received frame is a data frame 1 Received frame is a remote frame
R0 Bits 28:0 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31 ANMF: Accepted Non-matching Frame	Acceptance of non-matching frames may be enabled via GFC[ANFS] and GFC[ANFE]. 0 Received frame matching filter index FIDX 1 Received frame did not match any Rx filter element
R1 Bits 30:24 FIDX[6:0]: Filter Index	0-127=Index of matching Rx acceptance filter element (invalid if ANMF = ‘1’). Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
R1 Bit 21 EDL Extended Data Length	0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
R1 Bit 20 BRS Bit Rate Switch	0 Frame received without bit rate switching 1 Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]: Data Length Code	0-8 Received frame has 0-8 data bytes 9-15 Received frame has 8 data bytes
R1 Bits 15:0 RXTS[15:0]: Rx Timestamp	Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R2 Bits 31:24	DB3[7:0]: Data Byte 3
R2 Bits 23:16	DB2[7:0]: Data Byte 2
R2 Bits 15:8	DB1[7:0]: Data Byte 1
R2 Bits 7:0	DB0[7:0]: Data Byte 0
R3 Bits 31:24	DB7[7:0]: Data Byte 7
R3 Bits 23:16	DB6[7:0]: Data Byte 6
R3 Bits 15:8	DB5[7:0]: Data Byte 5
R3 Bits 7:0	DB4[7:0]: Data Byte 4

44.3.6.2 Tx buffer element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes

between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TXBC[TFQS] and TXBC[NDTB].

Figure 528. Tx Buffer Element

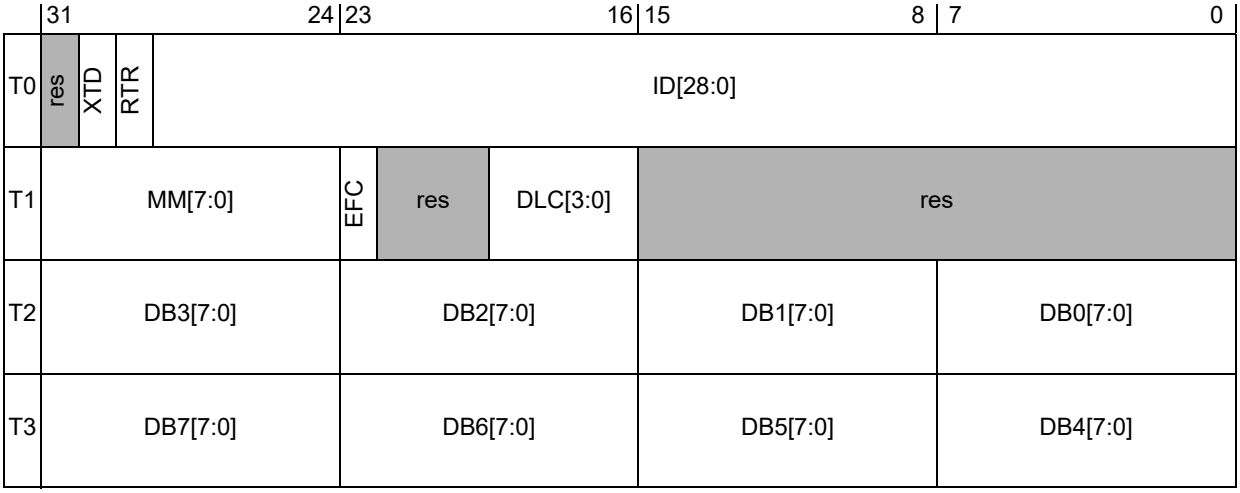


Table 590. Tx Buffer Element descriptions

T0 Bit 30 XTD: Extended Identifier	0 11-bit standard identifier 1 29-bit extended identifier
T0 Bit 29 RTR: Remote Transmission Request	0 Transmit data frame 1 Transmit remote frame
T0 Bits 28:0 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]: Message Marker	Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.
T1 Bit 23 EFC	Event FIFO Control 0 Don’t store Tx events 1 Store Tx events
T1 Bits 19:16 DLC[3:0]	Data Length Code 0-8 Transmit frame with 0-8 data bytes 9-15 Transmit frame with 8 data bytes
T2 Bits 31:24	DB3[7:0]: Data Byte 3
T2 Bits 23:16	DB2[7:0]: Data Byte 2
T2 Bits 15:8	DB1[7:0]: Data Byte 1
T2 Bits 7:0	DB0[7:0]: Data Byte 0
T3 Bits 31:24	DB7[7:0]: Data Byte 7
T3 Bits 23:16	DB6[7:0]: Data Byte 6

Table 590. Tx Buffer Element descriptions(Continued)

T3 Bits 15:8	DB5[7:0]: Data Byte 5
RT3 Bits 7:0	DB4[7:0]: Data Byte 4

44.3.6.3 Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

Figure 529. Tx Event FIFO Element

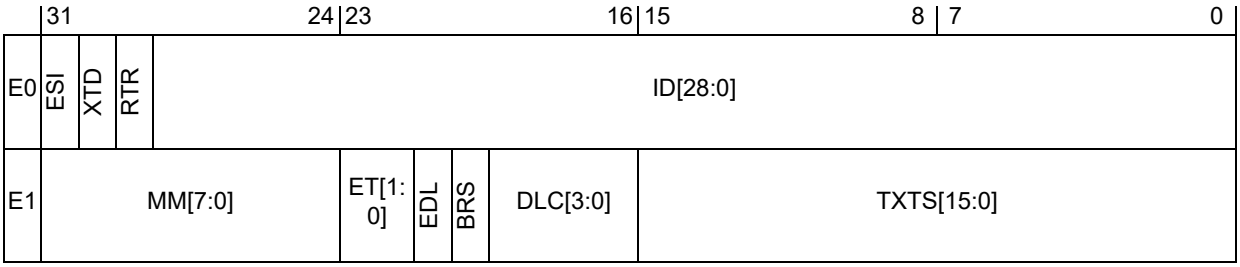


Table 591. Tx Event FIFO Element

E0 Bit 31 ESI	Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
E0 Bit 30 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
E0 Bit 29 RTR	Remote Transmission Request 0 Data frame transmitted 1 Remote frame transmitted
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
E1 Bits 31:24 MM[7:0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status
E1 Bit 23:22 ET[1:0]	Event Type 00 Reserved 01 Tx event 10 Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 Reserved
E1 Bit 21 EDL	Extended Data Length 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)

Table 591. Tx Event FIFO Element(Continued)

E1 Bit 20 BRS	Bit Rate Switch 0 Frame transmitted without bit rate switching 1 Frame transmitted with bit rate switching
E1 Bits 19:16 DLC[3:0]	Data Length Code 0-8 Frame with 0-8 data bytes transmitted 9-15 Frame with 8 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].

44.3.6.4 Standard message ID filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFC[FLSSA] plus the index of the filter element (0 to 127).

Figure 530. Standard Message ID Filter Element

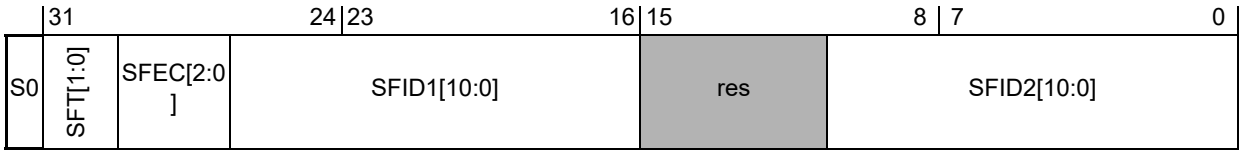


Table 592. Standard Message ID Filter Element Field Description

Bits 31:30 SFT[1:0]	Standard Filter Type 00 Range filter from SF1ID to SF2ID (SF2ID ≥ SF1ID) 01 Dual ID filter for SF1ID or SF2ID 10 Classic filter: SF1ID = filter, SF2ID = mask 11 Reserved
Bit 29:27 SFEC[2:0]	Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored

Table 592. Standard Message ID Filter Element Field Description(Continued)

Bits 26:16 SFID1[10:0]	Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 10:0 SFID2[10:0]	Standard Filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: 1) SFEC = “001”...”110” Second ID of standard ID filter element 2) SFEC = “111” Filter for Rx Buffers or for debug messages SFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C SFID2[8:6] is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one M_CAN host clock period in case the filter matches. SFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

Note: In case a reserved value is configured, the filter element is considered disabled.

44.3.6.5 Extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address XIDFC[FLESA] plus two times the index of the filter element (0…63).

Figure 531. Extended Message ID Filter Element

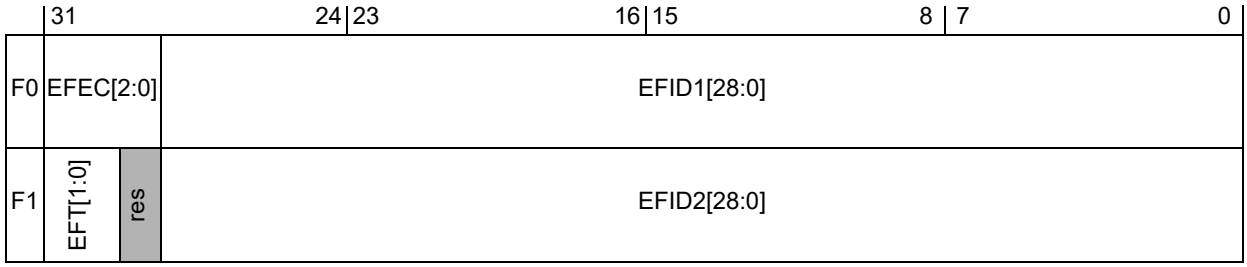


Table 593. Extended Message ID Filter Element Field Description

F0 Bits 31:29 EFEC[2:0]	Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = “100”, “101”, or “110” a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored
F0 Bits 28:0 EFID1[28:0]	Extended Filter ID 1 First ID of extended ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism (see *Section 44.3.6.5: Extended message ID filter element*) is used.
F1 Bits 31:30 EFT[1:0]	Extended Filter Type 00 Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) 01 Dual ID filter for EF1ID or EF2ID 10 Classic filter: EF1ID = filter, EF2ID = mask 11 Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), XIDAM mask not applied
F1 Bits 28:0 EFID2[28:0]	This bit field has a different meaning depending on the configuration of EFEC: 1) EFEC = “001”...”110” Second ID of extended ID filter element 2) EFEC = “111” Filter for Rx Buffers or for debug messages EFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C EFID2[8:6] is used to control the filter event pins m_can_fe[2:0] at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one m_can_hclk period in case the filter matches. EFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

Note: In case a reserved value is configured, the filter element is considered disabled.

44.3.7 M_CAN functional description

44.3.7.1 Operating modes

44.3.7.1.1 Software initialization

Software initialization is started by setting bit CCCR[INIT], either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While CCCR[INIT] is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus transmit output is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR[INIT] does not change any configuration register. Resetting CCCR[INIT] finishes the software initialization. Afterwards the Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits CCCR[INIT] and CCCR[CCE] are set (protected write).

CCCR[CCE] can only be set/reset while CCCR[INIT] = ‘1’. CCCR[CCE] is automatically reset when CCCR[INIT] is reset.

The following registers are reset when CCCR[CCE] is set

- HPMS–High Priority Message Status
- RXF0S–Rx FIFO 0 Status
- RXF1S–Rx FIFO 1 Status
- TXFQS–Tx FIFO/Queue Status
- TXBRP–Tx Buffer Request Pending
- TXBTO–Tx Buffer Transmission Occurred
- TXBCF–Tx Buffer Cancellation Finished
- TXEFS–Tx Event FIFO Status

The Timeout Counter value TOCV[TOC] is preset to the value configured by TOCC[TOP] when CCCR[CCE] is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while CCCR[CCE] = ‘1’.

The following registers are only writable while CCCR[CCE] = ‘0’

- TXBAR - Tx Buffer Add Request
- TXBCR - Tx Buffer Cancellation Request

CCCR[TEST] and CCCR.MON can only be set by the Host while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’. Both bits may be reset at any time. CCCR.DAR can only be set/reset while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’.

Following sequence can be used as an example code for initializing the CAN module (M_CAN0 in example):

```
-- Setting the INIT bit high to start the initialization
          while (read_data != 0x003){
            write(M_CAN0,0xF,driver.regs.can0_cccr.addr,0x003);
       read_data = read(M_CAN0,0xF,driver.regs.can0_cccr.addr);
          };
```

```
-- Configuring the time stamp and timeout registers
          write(M_CAN0,0xF,driver.regs.can0_tscc.addr,0x000);
          write(M_CAN0,0xF,driver.regs.can0_tocc.addr,0x000);

-- Configuring the interrupt registers
          write(M_CAN0,0xF,driver.regs.can0_ie.addr,0xFFFFFFFF);
          write(M_CAN0,0xF,driver.regs.can0_ils.addr,0x00000002);
          write(M_CAN0,0xF,driver.regs.can0_ile.addr,0x00000003);

          write(M_CAN0,0xF,driver.regs.can0_gfc.addr,0x000);
          write(M_CAN0,0xF,driver.regs.can0_xidam.addr,0x00FFFFFF);

--Writing the start addresses of all the M_CAN Buffers
          generate and drive M_CAN0 can0_rxf0c keeping {
                                                        .f0sa == 0x100;
                                                        .f0s  == 0x03;
                                                        .f0wm == 0x01;
                                                        };
          generate and drive M_CAN0 can0_rxf1c keeping {
                                                        .f1sa == 0x200;
                                                        .f1s  == 0x03;
                                                        .f1wm == 0x01;
                                                        };
          generate and drive M_CAN0 can0_sidfc keeping {
                                                        .flssa == 0x000;
                                                        .lss   == 0x80;
                                                        };
          generate and drive M_CAN0 can0_xidfc keeping {
                                                        .flesa == 0x080;
                                                        .lse   == 0x40;
                                                        };
          generate and drive M_CAN0 can0_txbc  keeping {
                                                        .tbsa == 0x300;
                                                        .ndtb == 0x020;
                                                        .tfqs == 0x1D;
                                                        };
          generate and drive M_CAN0 can0_txefc keeping {
                                                        .efsa == 0x380;
                                                        .efs == 0x020;
                                                        .efwm == 0x00;
                                                        };
--Writing to the RAM for filter configuration
================================================
--   Writing to TX Buffer
```

```
            write(RAM,0xF,0xC00,0x04040000);
            write(RAM,0xF,0xC04,0x10830000);
            write(RAM,0xF,0xC08,0x00010001);
            write(RAM,0xF,0xC10,0x50000002);
            write(RAM,0xF,0xC14,0x11830000);
            write(RAM,0xF,0xC18,0x00070102);

--    Writing to s-filter
            write(RAM,0xF,0x000,0x58020004);
            write(RAM,0xF,0x004,0x08000005);
            write(RAM,0xF,0x008,0x58060007);
            write(RAM,0xF,0x00C,0x08030008);
            write(RAM,0xF,0x010,0x880E07FB);
            write(RAM,0xF,0x014,0x980D07FE);
            write(RAM,0xF,0x018,0x68100100);
            write(RAM,0xF,0x01C,0x70120200);
            write(RAM,0xF,0x020,0x981F07F6);
            write(RAM,0xF,0x024,0x50140014);
            write(RAM,0xF,0x028,0x58160017);
            write(RAM,0xF,0x02C,0x08110019);
            write(RAM,0xF,0x030,0x4A22001A);
            write(RAM,0xF,0x034,0xA01A07FF);
            write(RAM,0xF,0x038,0x5BFF07E3);
            write(RAM,0xF,0x03C,0x080007FF);

--    Writing to x-filter
            write(RAM,0xF,0x200,0x201ABCD9);
            write(RAM,0xF,0x204,0x001ABCE5);
            write(RAM,0xF,0x208,0x403BCDEF);
            write(RAM,0xF,0x20C,0x003BCDEF);
            write(RAM,0xF,0x210,0x605CDEF2);
            write(RAM,0xF,0x214,0x405CDEF0);
            write(RAM,0xF,0x218,0x407DEEFF);
            write(RAM,0xF,0x21C,0x007DEF01);
            write(RAM,0xF,0x220,0x209ABCDE);
            write(RAM,0xF,0x224,0x009ABCDF);
            write(RAM,0xF,0x228,0xA0BBCDEF);
            write(RAM,0xF,0x22C,0x40BBCDEE);
            write(RAM,0xF,0x230,0x20DCDEF0);
            write(RAM,0xF,0x234,0x9FFFFFFF);
            write(RAM,0xF,0x238,0x40000000);
            write(RAM,0xF,0x23C,0x1FFFFFFF);
================================================
write(M_CAN0,0xF,driver.regs.can0_btp.addr,0x00002303);
```

```
    print "Configuration done";
    read_data = 0x01;
    while (read_data != 0x00){
      write(M_CAN0,0xF,driver.regs.can0_cccr.addr, 0x000);
wait [1];
read_data = read(M_CAN0,0xF,driver.regs.can0_cccr.addr);
    };

    print "Initialization done";
```

44.3.7.2 Normal operation

Once the M_CAN is initialized and CCCR[INIT] is reset to zero, the M_CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

44.3.7.3 CAN FD Operation

There are two stages in the CAN FD protocol, first the Long Frame Mode where the data field of a CAN frame may be longer that 8 bytes. The second stage is the Fast Frame Mode where control field, data field, and CRC field of a CAN Frame are transmitted with a higher bit rate than the beginning and the end of the frame. Fast Frame Mode can only be used in combination with Long Frame Mode.

The CAN operation mode is chosen by programming CCCR.CME. In case CCCR.CME = g01h transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CCCR.CME = g10h/g11h transmission and reception of long and fast CAN FD frames is enabled. CCCR.CME can only be changed while CCCR.INIT and CCCR.CCE are both set.

When initialization is left (CCCR.INIT set to e0f), both Long Frame Mode and Fast Frame Mode are inactive, they have to be requested by writing to CCCR.CMR. A mode change requested by writing to CCCR.CMR will be executed next time the CAN protocol controller FSM reaches idle phase between CAN frames. Upon this event CCCR.CMR is reset to g00h and the status flags CCCR.FACT and CCCR.LACT are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CCCR.CMR is retained until it is overwritten by the next mode change request. Default is normal CAN operation. A change of the CAN operation mode should not be requested while there are pending transmission requests.

When CCCR.CME , g00h, received frames are interpreted according to the CAN FD Protocol Specification. The reserved bit in CAN frames with 11-bit identifiers and the first reserved bit in CAN frames with 29-bit identifiers will be decoded as EDL bit. EDL = recessive signifies a CAN FD long frame, EDL = dominant signifies a standard CAN frame. In a CAN FD long frame, the two bits following EDL, r0 and BRS, decide whether this CAN FD long frame is also a CAN FD fast frame. A CAN FD fast frame is signified by r0 = dominant and BRS = recessive. The coding of r0 = recessive is reserved for future expansion of the protocol.

The status bits CCCR.LACT and CCCR.FACT indicate the format of transmitted frames. When CCCR.LACT is set, frames will be transmitted in CAN FD long format with EDL = recessive. When both CCCR.LACT and CCCR.FACT are set, frames will be transmitted in CAN FD Fast format with both EDL and BRS = recessive.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to the following table.

Table 594. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing and Prescaler Register BTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the Fast Bit Timing and Prescaler The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames the ESI bit is always transmitted dominant, independent of the transmitter's error state. The data length code of CAN FD remote frames is transmitted as zero.

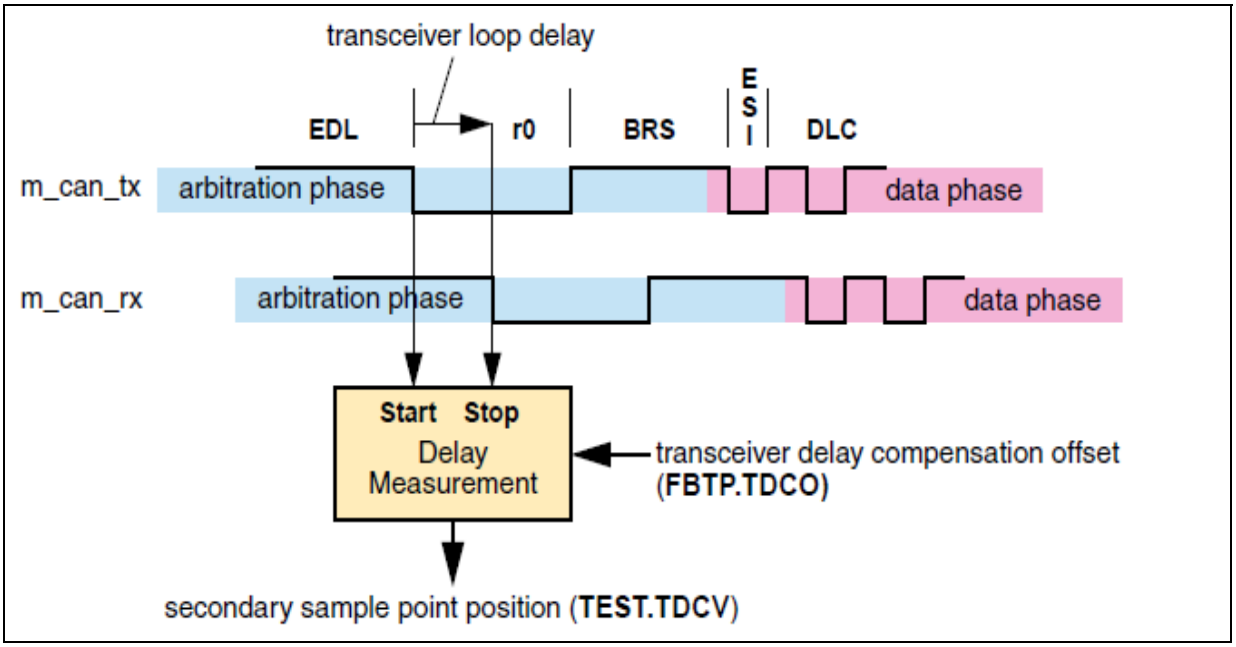
In case a M_CAN Tx Buffer is configured for CAN FD transmission with DLC > 8, the first 8 bytes are transmitted as configured in the Tx Buffer while the remaining part of the data field is padded with 0xCC. When the M_CAN receives a CAN FD frame with DLC > 8, the first 8 bytes of that frame are stored into the matching Rx Buffer or Rx FIFO. The remaining bytes are discarded.

44.3.7.4 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin m_can_tx the protocol controller receives the transmitted data from its local CAN transceiver via pin m_can_rx. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver. Figure 532 below describes how the transceiver loop delay is measured.

Figure 532. Transceiver delay measurement



Within each CAN FD frame, the transmitter measures the delay between the data transmitted at pin m_can_tx and the data received at pin m_can_rx, starting with the falling edge of bit EDL. The delay is measured in m_can_cclk periods.

A secondary sample point is calculated by adding a configurable transceiver delay compensation offset (FBTP.TDCO) to the measured transceiver delay. The transceiver delay compensation offset is used to adjust the secondary sample point inside the bit time (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of time quanta tq.

To check for bit errors during the data phase, the delayed transmit data is compared against the received data at the secondary sample point. If a bit error is detected at the secondary sample point, the transmitter will react to this bit error at the next following sample point. During arbitration phase the delay compensation is always disabled.

The maximum delay which can be compensated by the M_CAN's delay compensation during the data phase is three bit times.

44.3.7.4.1 Configuration and status

Compensation for the transceiver loop delay by the M_CAN is enabled via FBTP.TDC. The transceiver delay compensation offset is configured via FBTP.TDCO. The actual delay compensation value applied by the M_CAN's protocol engine can be read from TEST.TDCV.

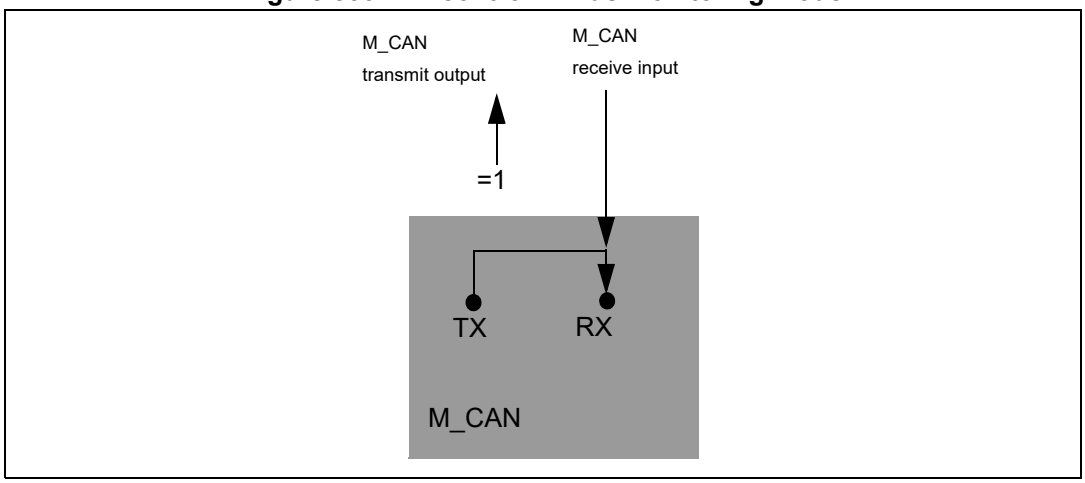
44.3.7.5 Bus monitoring mode

The M_CAN is set in Bus Monitoring Mode by programming CCCR[MON] to one. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the M_CAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the M_CAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_CAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode, register TXBRP is held in reset state. The Bus Monitoring Mode can be

used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits.

Figure 533 shows the connection of M_CAN Tx and Rx signals to the M_CAN in Bus Monitoring Mode.

Figure 533. Pin control in Bus Monitoring mode



44.3.7.6 Disabled automatic retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the M_CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via CCCR[DAR].

44.3.7.6.1 Frame transmission in DAR mode

In DAR mode, all transmissions are automatically canceled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit TXBRP[TRPx] is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] not set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = "10" (transmission in spite of cancellation).

44.3.7.7 Power down (Sleep mode)

The M_CAN can be set into power down mode controlled by input signal clock stop request or via CC Control Register CCCR[CSR]. As long as the clock stop request signal is active, bit CCCR[CSR] is read as one. When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the M_CAN sets then CCCR[INIT] to one to prevent any further CAN transfers. Now the M_CAN acknowledges that it is ready for power down by setting clock stop acknowledge output signal to one and CCCR[CSA] to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] will have no effect. Now the module clock inputs (CAN clock and host clock) may be switched off. To leave power down mode, the application has to turn on the module clocks before resetting signal clock stop request signal resp. CC Control Register flag CCCR[CSR]. The M_CAN will acknowledge this by resetting clock stop acknowledge output signal and resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

44.3.7.8 Test modes

To enable write access to register TEST (see *Section 44.3.5.2.4: Test Register*), bit CCCR[TEST] has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin m_can_tx by programming TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_CAN's bit timing and it can drive constant dominant or recessive values. The actual value at M_CAN Rx pin can be read from TEST[RX]. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to TEST[TX] until the new configuration is visible at output Tx pin. This applies also when reading input Rx pin via TEST[RX].

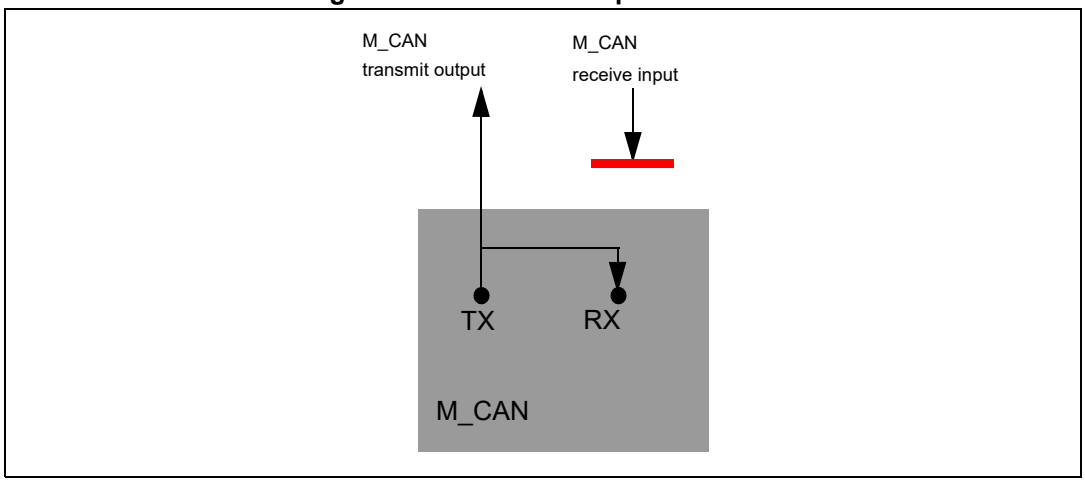
Note: Test modes should be used for production tests or self test only. The software control for Tx pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

44.3.7.8.1 External Loop Back mode

The M_CAN can be set in External Loop Back mode by programming TEST.LBCK to one. In Loop Back mode, the M_CAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 534* shows the connection of Tx and Rx signals to the M_CAN in External Loop Back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back mode. In this mode the M_CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN Rx input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the M_CAN Tx pin.

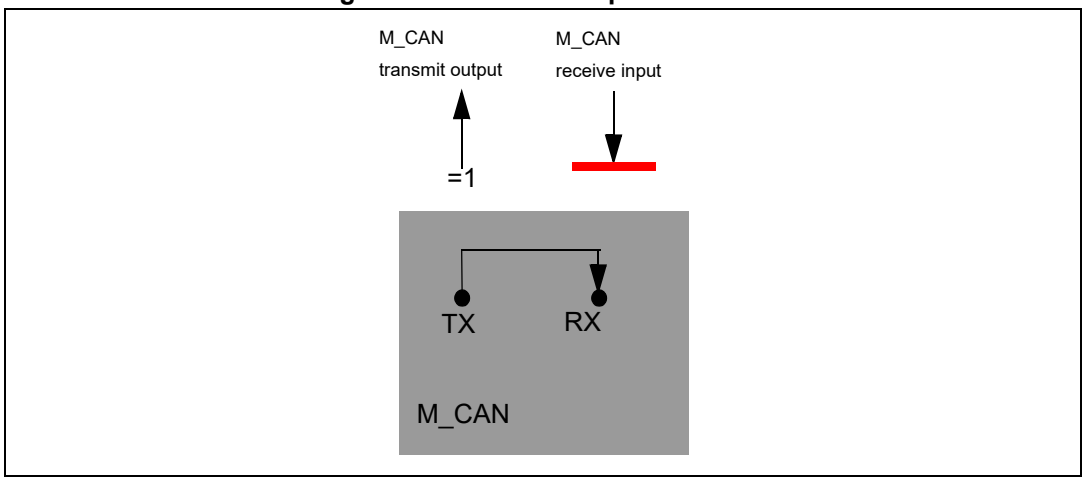
Figure 534. External Loop Back mode



44.3.7.8.2 Internal Loop Back mode

Internal Loop Back mode is entered by programming bits TEST[LBCK] and CCCR[MON] to one. This mode can be used for a “Hot Selftest”, meaning the M_CAN can be tested without affecting a running CAN system connected to the M_CAN Tx and Rx pins. In this mode M_CAN Tx pin is disconnected from the M_CAN and M_CAN Tx pin is held recessive. *Figure 535* shows the connection of M_CAN Tx pin and Rx to the M_CAN in case of Internal Loop Back mode.

Figure 535. Internal loop back mode



44.3.8 Timestamp generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler TSCC[TCP] can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via TSCV[TSC]. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer /Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit TSCC.TSS, a 16-bit timestamp can be used.

44.3.9 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_CAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC]. The Timeout Counter can only be started while CCCR[INIT] = '0'. It is stopped when CCCR[INIT] = '1', e.g. when the M_CAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

Note: The clock signal for the Timeout Counter is derived from the CAN core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core.If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

44.3.10 Rx handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

44.3.10.1 Acceptance filtering

The M_CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (GFC)
- Standard ID Filter Configuration (SIDFC)
- Extended ID Filter Configuration (XIDFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Reject received frame
- Set High Priority Message interrupt flag IR[HPM]
- Set High Priority Message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1

Note: When an accepted message is written to one of the two Rx FIFOs, or into a Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

44.3.10.1.1 Range filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID. There are two possibilities when range filtering is used together with extended frames:

- EFT = “00”: The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied
- EFT = “11”: The Extended ID AND Mask (XIDAM) is not used for range filtering

44.3.10.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID =EF2ID.

44.3.10.1.3 Filter for dedicated IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

44.3.10.1.4 Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering. In case all mask bits are one, a

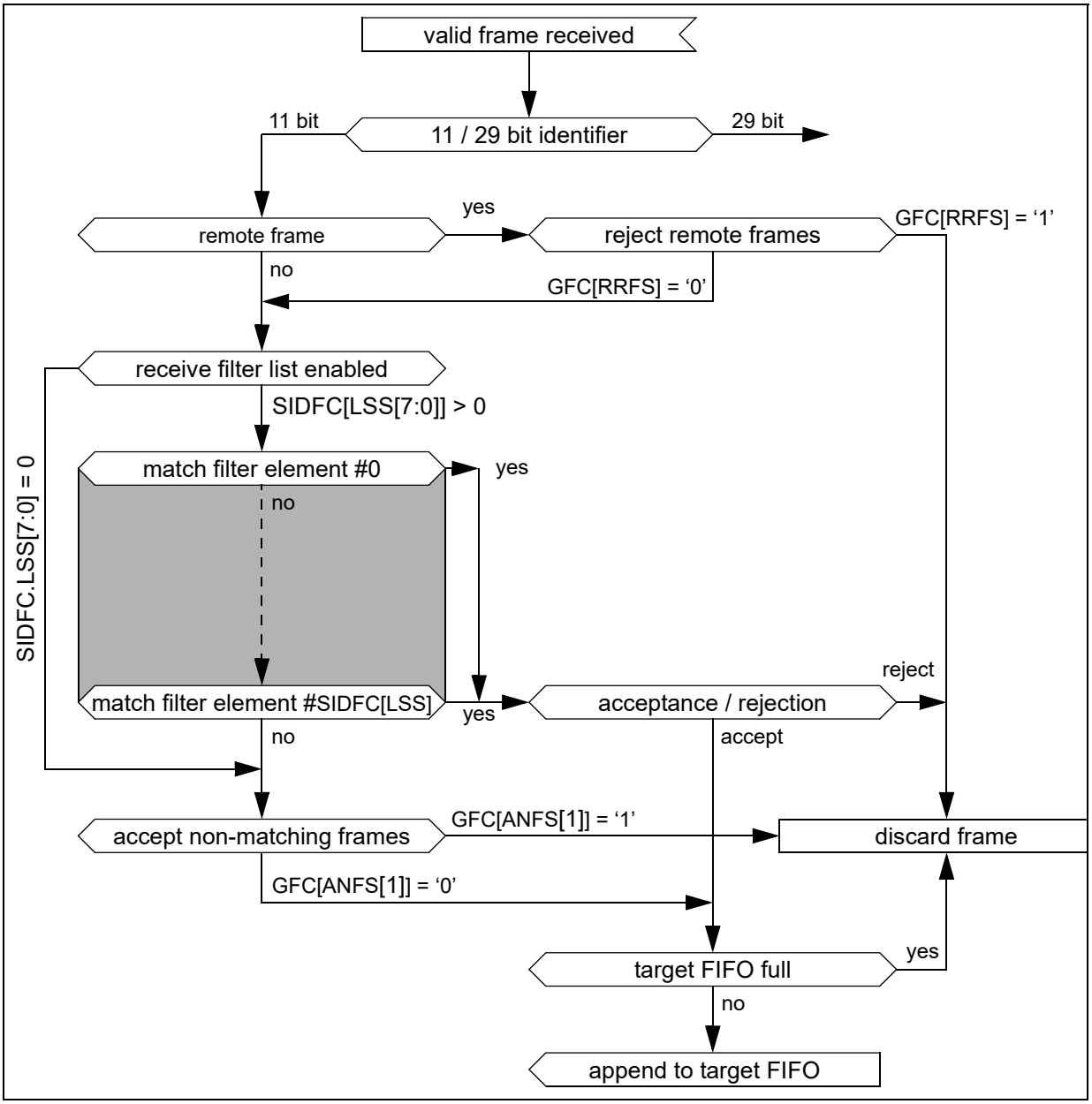
match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

44.3.10.1.5 Standard message ID filtering

Figure 536 shows the flow for standard message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in *Section 44.3.6.4: Standard message ID filter element*.

Controlled by the Global Filter Configuration GFC and the Standard ID Filter Configuration SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Figure 536. Standard Message ID Filter Path

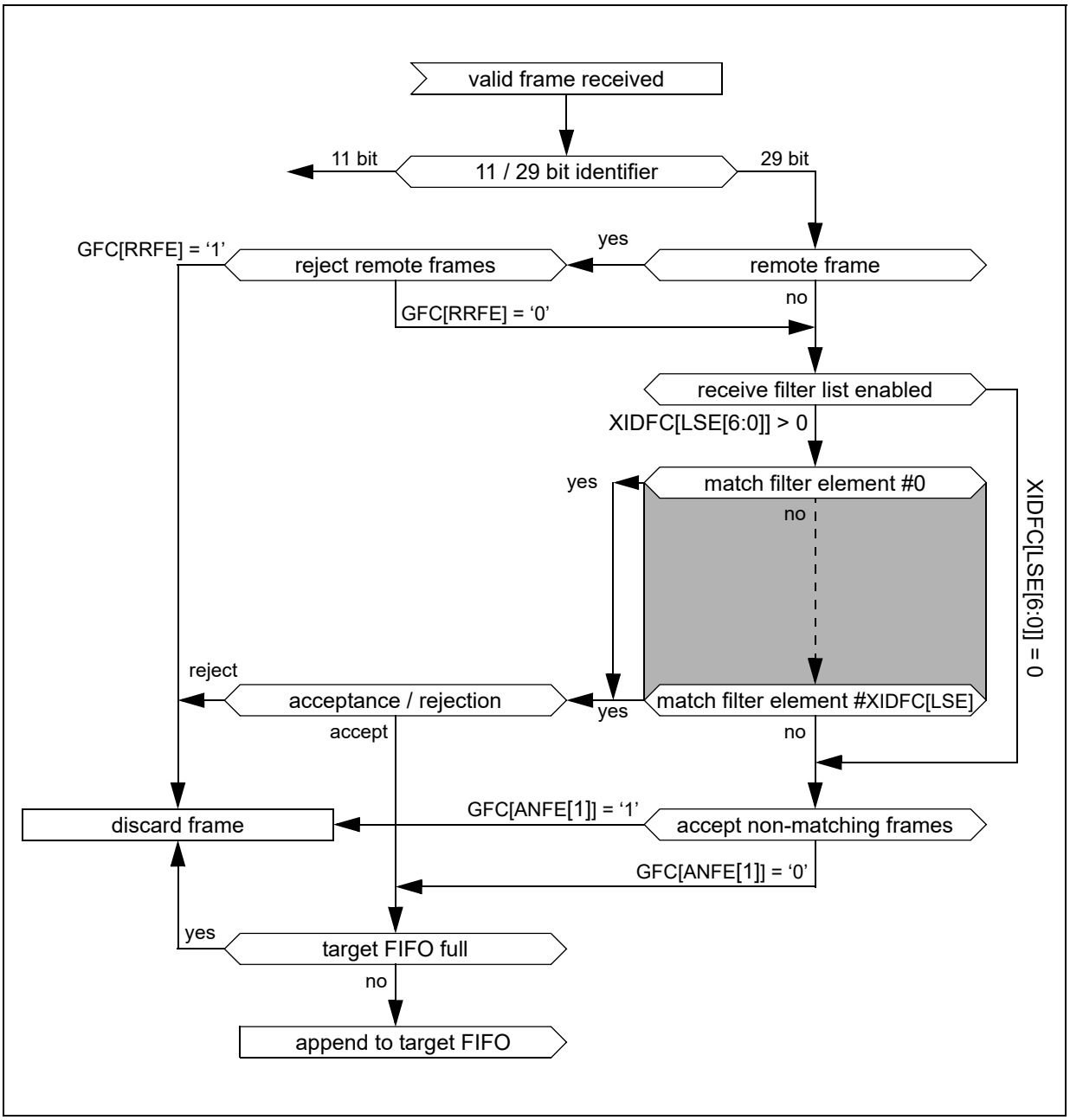


44.3.10.1.6 Extended message ID filtering

Figure 537 below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in *Section 44.3.5.2.20: Extended ID Filter Configuration Register (XIDFC)*.

Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements which is controlled by the Global Filter Configuration (GFC) and the Extended ID Filter Configuration (XIDFC) Message ID. The Extended ID AND Mask (XIDAM) is ANDed with the received identifier before the filter list is executed.

Figure 537. Extended Message ID Filter Path



44.3.10.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C. Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see *Section 44.3.10.1: Acceptance filtering*. The Rx FIFO element is described in *Section 44.3.6.1: Rx Buffer and FIFO element*.

When an Rx FIFO full condition is signaled by IR[RFnF], no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded and interrupt flag IR.RFnL is set. To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC[FnWM], interrupt flag IR[RFnW] is set. When reading from an Rx FIFO, four times the Rx FIFO Get Index RXFnS[FnGI] has to be added to the corresponding Rx FIFO start address RXFnC[FnSA].

44.3.11 Dedicated Rx Buffers

The M_CAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = “111” and SFID2 / EFID2[10:9] = “00” has to be configured (see *Section 44.3.6.4: Standard message ID filter element* and *Section 44.3.6.5: Extended message ID filter element*).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Table 595. Example Filter Configuration for Rx buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message 1	00	00 0000
1	ID debug message 2	00	00 0001
2	ID debug message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a ‘1’ to the respective bit position.

While an Rx Buffer’s New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

44.3.11.1 Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

44.3.12 Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (e.g.#61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (see *Section 44.3.6.1: Rx Buffer and FIFO element*).

Advantage: Fixed start address for the DMA transfers (relative to RXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = “111” have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output m_can_dma_req is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M_CAN while m_can_dma_req is activated. The behavior is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets m_can_dma_ack. This resets m_can_dma_req. Now the M_CAN is prepared to receive the next set of debug messages.

44.3.12.1 Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to “111”. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 /EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor IR.DRX are set. The reception of debug messages can be monitored via RXF1S.DMS.

Table 596. Example Filter Configuration for debug messages

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

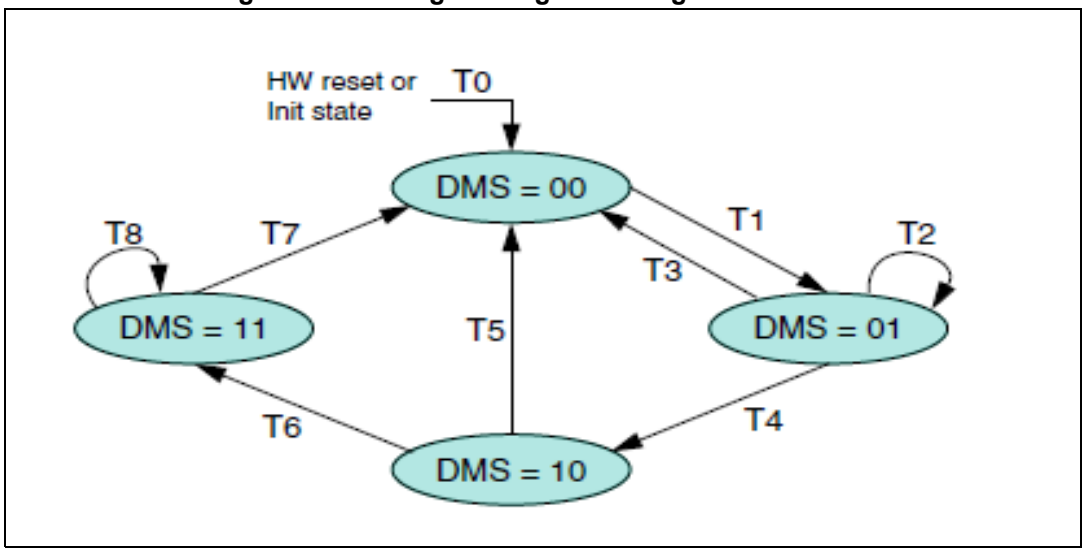
44.3.12.2 Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is

restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signaled via RXF1S.DMS.

Figure 538. Debug Message Handling State Machine



T0: reset m_cam_dma_req output, enable reception of debug messages A, B, and C

T1: reception of debug message A

T2: reception of debug message A

T3: reception of debug message C

T4: reception of debug message B

T5: reception of debug messages A, B

T6: reception of debug message C

T7: DMA transfer completed

T8: reception of debug message A,B,C (message rejected)

44.3.13 Tx handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in *Section 44.3.6.2: Tx buffer element*.

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

44.3.13.1 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via TXBAR[ARn]. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates four 32-bit words in the Message RAM. Therefore the start address of a Dedicated Tx Buffer in the Message RAM is calculated by adding four times the transmit buffer index (0…31) to the Tx Buffer Start Address TXBC[TBSA].

44.3.13.2 Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to ‘0’. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_CAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF] = ‘1’) is signaled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a ‘1’ to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFOs Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.When a transmission request for the Tx Buffer referenced by the Get Index is canceled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding four times the Put Index TXFQS[TFQPI] (0…31) to the Tx Buffer Start Address TXBC[TBSA].

44.3.13.3 Tx Queue

Tx Queue operation is configured by programming TXBC[TFQM] to ‘1’. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first. New messages have to be written to the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TXFQS[TFQF] = ‘1’), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

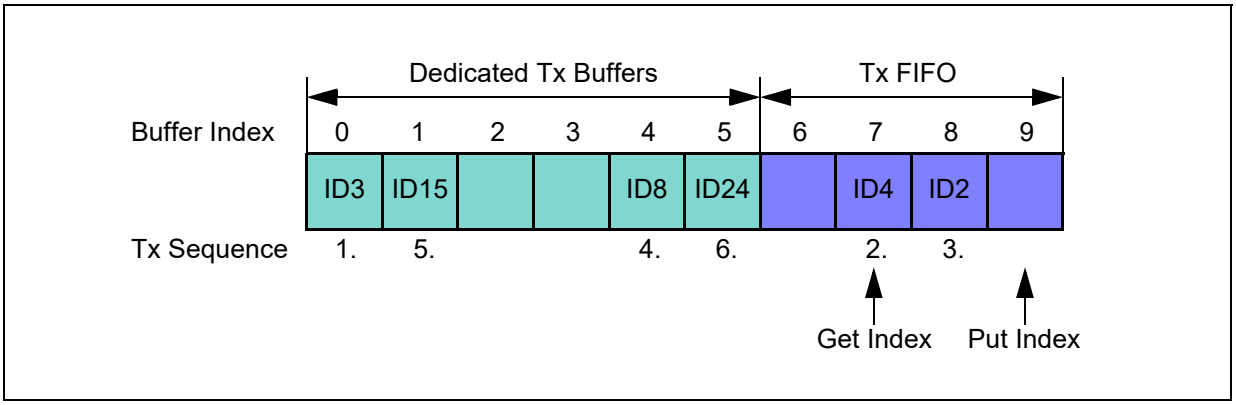
The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding four times the Tx Queue Put Index TXFQS[TFQPI] (0…31) to the Tx Buffer Start Address TXBC[TBSA].

44.3.13.4 Mixed dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Buffers assigned to the Tx FIFO is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

Figure 539. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO



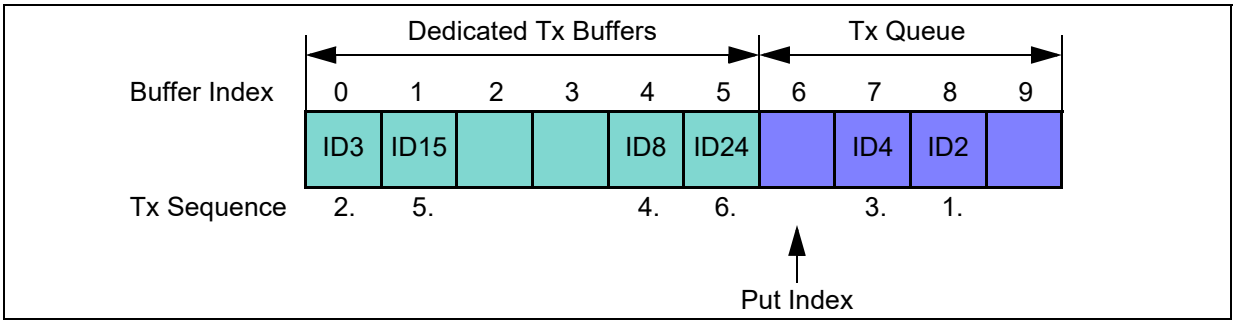
Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS[TFGI])
- Buffer with lowest Message ID gets highest priority and is transmitted next

44.3.13.5 Mixed dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Queue Buffers is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

Figure 540. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue



Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

44.3.13.6 Transmit cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register TXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

44.3.13.7 Tx Event handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in *Section 44.3.6.3: Tx event FIFO element*. When a Tx Event FIFO full condition is signaled by IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC[EFWM], interrupt flag IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS[EFGI] has to be added to the Tx Event FIFO start address TXEFC[EFSA].

44.3.14 FIFO acknowledge handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see *Section 44.3.5.2.27: Rx FIFO 0 Acknowledge Register (RXF0A)*, *Section 44.3.5.2.31: Rx FIFO 1 Acknowledge Register (RXF1A)*, and *Section 44.3.5.2.43: Tx Event FIFO Acknowledge Register (TXEFA)*). Writing

to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFOs Get Index.

Due to the fact that the CPU has free access to the M_CAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFOs Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFOs Fill Level. In this case some of the older FIFO elements would be lost.

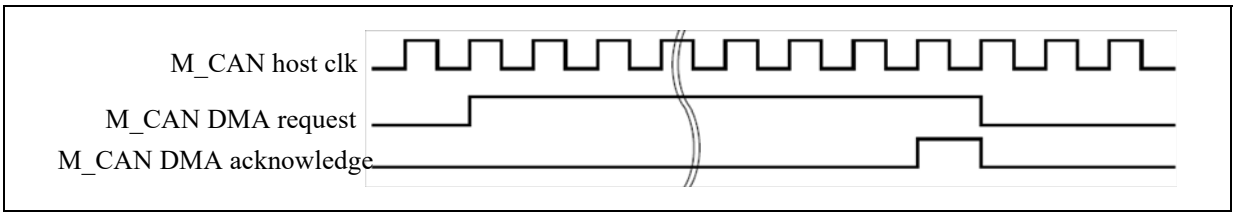
Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

44.3.15 Interface to DMA Controller

When all three debug messages A, B, C have been received in the correct order, M_CAN DMA request signal is activated to trigger a DMA transfer. The RAM words holding debug messages A, B, C will not be changed by the M_CAN while M_CAN DMA request signal is active.

After the transfer of the received messages has completed the DMA unit activates the M_CAN DMA ACK signal. This resets M_CAN DMA request signal. The debug message handling state machine enters idle state (DMS = "00") and waits for reception of the next debug messages.

Figure 541. Timing of DMA interface signals



Note: If the DMA unit activates input M_CAN DMA acknowledge signal before the DMA transfer has completed, the Rx Buffer elements holding debug messages A, B, C are unlocked and may be overwritten by received debug messages.

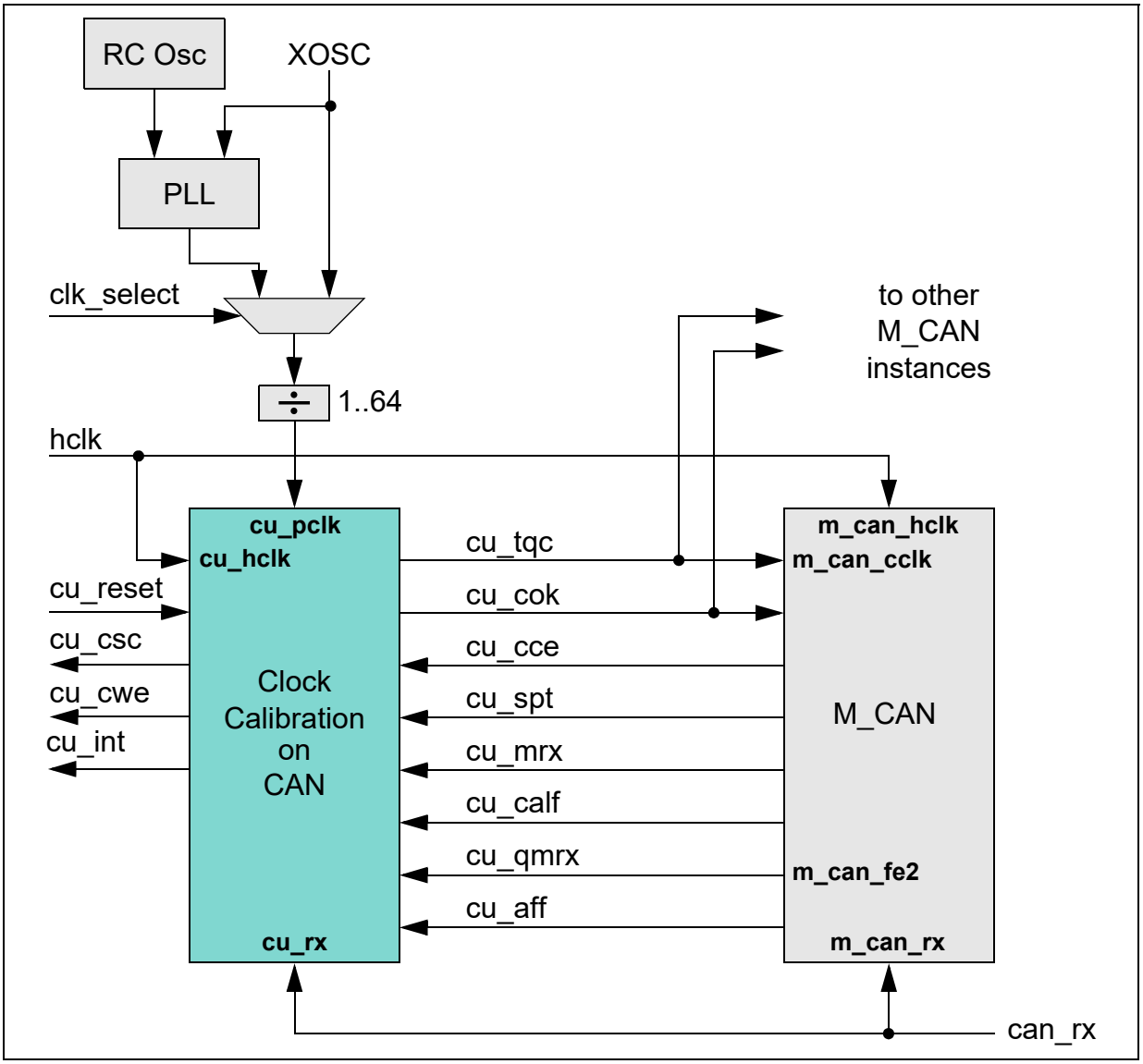
44.3.16 Clock Calibration on M_CAN

This section describes an add-on for the M_CAN modules that generates a calibrated CAN clock **cu_tqc** from an on-chip RC-oscillator clock by evaluating CAN messages received by the attached M_CAN. Calibration on CAN is only possible when the M_CAN is operated in standard CAN mode.

Figure 542 shows the interface of the Clock Calibration on CAN unit. The clock multiplexer controlled by **clk_select** controls whether the clock from the PLL or XOSC is routed to clock

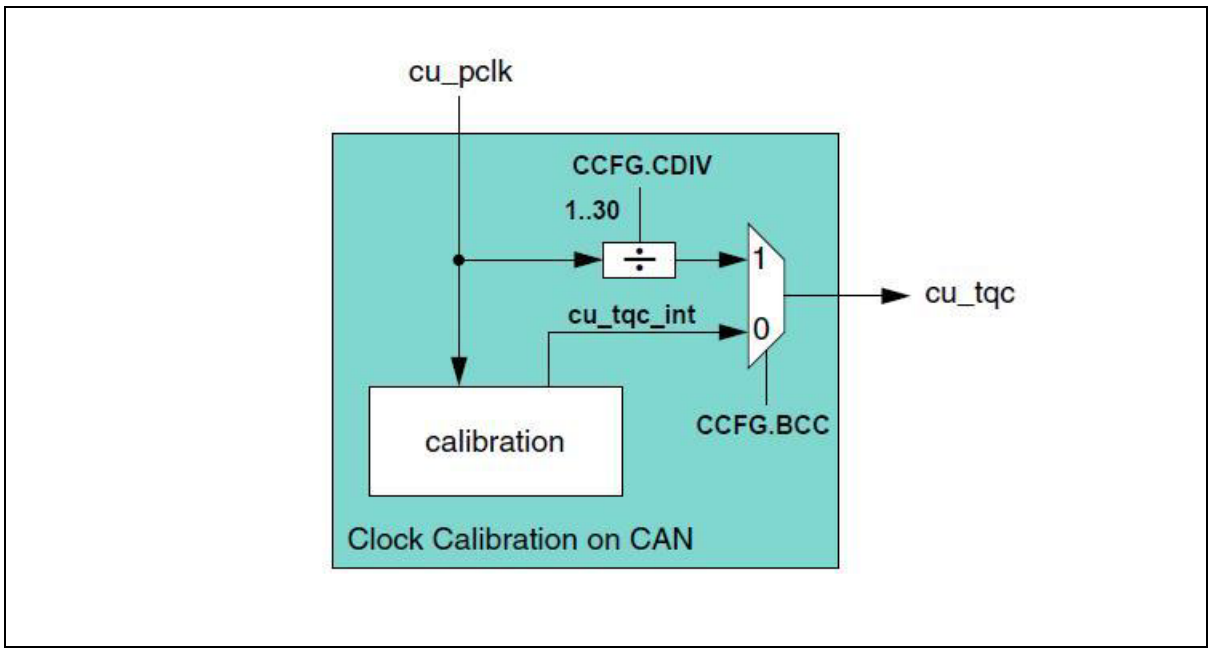
input **cu_pclk**. The calibrated time quanta clock **cu_tqc** is routed to all CAN instances. In case of a successful calibration this is signaled via **cu_cok**.

Figure 542. Connection of Clock Calibration on CAN unit to M_CAN



CPU accesses to the Clock Calibration on CAN unit use the Host clock connected to **cu_hclk**. Clock calibration is bypassed when CCFG.BCC = '1'. In this case, the clock from input **cu_pclk** is routed through a clock divider to output **cu_tqc** (see *Figure 543*).

Figure 543. Bypass operation



For details of signals refer to *Table 605: Calibration on CAN Module Interface*

44.3.16.1 Operating Conditions

The Clock Calibration on CAN unit is designed to operate under the following conditions:

- Clock input from PLL (**cu_pclk**) between 80 MHz and 500 MHz
- M_CAN bit rates between 125 kbit/s and 1 Mbit/s
- Clock for Host accesses (**cu_hclk**) is 40/50 MHz

The Clock Calibration on CAN unit generates a calibrated time quanta clock **cu_tqc** in the range from 0.5 MHz to 25 MHz.

Note: The M_CAN requires that the CAN clock is always below or equal to the Host clock (m_can_cclk ≤ m_can_hclk). This has to be considered when the Clock Calibration on CAN unit is bypassed (CCFG.BCC = '1').

44.3.16.2 Calibration Accuracy

The calibration accuracy in state Precision_Calibrated depends on:

- The dynamic clock tolerance at the clock input **cu_pclk**
- The measurement error. For each bit sequence used for calibration measurement, there is a maximum error of one **cu_pclk** period. The number of bits used for measurement of the bit time is 32 or 64-bit, depending on configuration of CCFG.CFL.
- Tolerable error in calibration mechanism

The distance between two calibration messages has to be chosen to fit the clock tolerance requirements of the attached M_CANs.

Note: Dynamic clock tolerance is the clock frequency variation between two calibration messages e.g. caused by change of temperature or operating voltage.

44.3.16.3 Memory map and register description

44.3.16.3.1 Hardware Reset Description

After hardware reset, the registers of the Clock Calibration on CAN unit hold the reset values listed in Table .

The Clock Calibration on CAN unit allocates an address space of 16 bytes. All registers are organized as 32-bit registers. The Clock Calibration on CAN unit is accessible by the Host CPU using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access). Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when input **cu_cce** = ‘1’. Signal **cu_cce** is activated when the M_CAN control bits CCCR.CCE = ‘1’ AND CCCR.INIT = ‘1’.

Table 597. Clock Calibration on CAN Register Map

Address Offset	Register Name	Access (1)	Reset value	Section/Page
0x0	Core Release Register (CREL)	R	rrrd dddd	*on page 1039*
0x1	Calibration Configuration Register (CCFG)	RP	0000 0404	*on page 1040*
0x2	Calibration Status Register (CSTAT)	R	0203 FFFF	*on page 1042*
0x3	Calibration Watchdog (CWD)	RP	0000 0000	*on page 1042*
0x4	CU Interrupt Register (CUIR)	W1C	0000 0000	*on page 1043*
0x5	CU Interrupt Enable (CUIE)	RW	0000 0000	*on page 1044*

1. R = Read, W = Write, P = Protected Write, r = release, d = date.

44.3.16.3.2 Core Release Register (CREL)

Address: 0x0 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-d	-d	-d	-d

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MON								DAY							
W																
Reset	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d

R = Read; -r = release, -d = time stamp, the coding of revisions depends on the module version used in the device

Figure 544. Core Release Register (CREL)

Table 598. Tx Core Release Register field descriptions

Field	Description
0:3 REL	Core Release One digit, BCD-coded.
4:7 STEP	Step of Core Release One digit, BCD-coded.
8:11 SUBSTEP	Sub-step of Core Release One digit, BCD-coded.
12:15 YEAR	Time Stamp Year One digit, BCD-coded.
12:15 MON	Time Stamp Month Two digits, BCD-coded.
12:15 DAY	Time Stamp Day Two digits, BCD-coded.

Table 599. Example for Coding of Revisions

Release	Step	SubStep	Year	Month	Day	Name
1	0	0	2	02	20	Revision 1.0.0, Date 2012/02/20

44.3.16.3.3 Calibration Configuration Register (CCFG)

Address: 0x1 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWR	0	0	0	0	0	0	0	0	0	0	0	CDIV(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OCPM(1)								CFL (1)	BCC (1)	0	TQBT(1)				
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Figure 545. Calibration Configuration Register (CCFG)

1. Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when input cu_cce = ‘1’. Signal cu_cce is activated when the M_CAN control bits CCCR.CCE = ‘1’ AND CCCR.INIT = ‘1’.

Table 600. Calibration Configuration Register field descriptions

Field	Description
0 SWR	Software Reset Writing a ‘1’ to this bit will reset the calibration FSM to state Not_Calibrated (CSTAT.CALS = “00”). The Calibration Watchdog value CWD.WDV is also reset. Registers CCFG, CSTAT and the Calibration Watchdog configuration CWD.WDC are unchanged. The bit remains set until reset has completed.
1:11	Reserved
12:15 CDIV	Clock Divider The clock divider has to be configured when the clock calibration is bypassed (BCC = ‘1’) to assure that the M_CAN requirement m_can_cclk . m_can_hclk is fulfilled. 0000 Divide by 1 0001 Divide by 2 0010 Divide by 4 0011 Divide by 6 0100 Divide by 8 0101 Divide by 10 0110 Divide by 12 0111 Divide by 14 1000 Divide by 16 1001 Divide by 18 1010 Divide by 20 1011 Divide by 22 1100 Divide by 24 1101 Divide by 26 1110 Divide by 28 1111 Divide by 30
16:23 OCPM	Oscillator Clock Periods Minimum Configures the minimum number of cu_pclk periods in two CAN bit times. OCPM is used in Basic Calibration to avoid false measurements in case of glitches on the bus line. The configured number of cu_pclk periods is OCPM • 32. The configuration depends on the cu_pclk frequency (80 MHz to 500 MHz) and the bit rate configured in the attached M_CANs (125 kbit/s to 1 Mbit/s). It is recommended to configure a value slightly below two CAN bit times. The reset value is 1. 6 bit times at 80 MHz cu_pclk and 1 Mbit/s CAN bit rate.
24 CFL	Calibration Field Length 0 Calibration field length is 32 bits 1 Calibration field length is 64 bits
25 BCC	Bypass Clock Calibration If this bit is set, the clock input cu_pclk is routed to the time quanta clock output cu_tqc through a clock divider configurable via CDIV, cu_cok is always ‘1’. In this case the baud rate prescaler of the connected M_CANs has to be configured to generate the M_CAN internal time quanta clock. 0 Clock calibration unit generates time quanta clock cu_tqc 1 Clock calibration unit bypassed **Note:** As long as cu_pclk is equal or above 80 MHz the Clock Calibration on CAN unit is functional, even when BCC = ‘1’. The calibration state can be read from register CSTAT.
26	Reserved
27:31 TQBT	Time Quanta per Bit Time Configures the number of time quanta per bit time. Same value as configured in the attached M_CANs. The range of the resulting time quanta clock cu_tqc is from 0,5 MHz (bit rate of 125 kbit/s with 4 tq per bit time) to 25 MHz (bit rate of 1 Mbit/s with 25 tq per bit time). Valid values are 4 to 25.

44.3.16.3.4 Calibration Status Register (CSTAT)

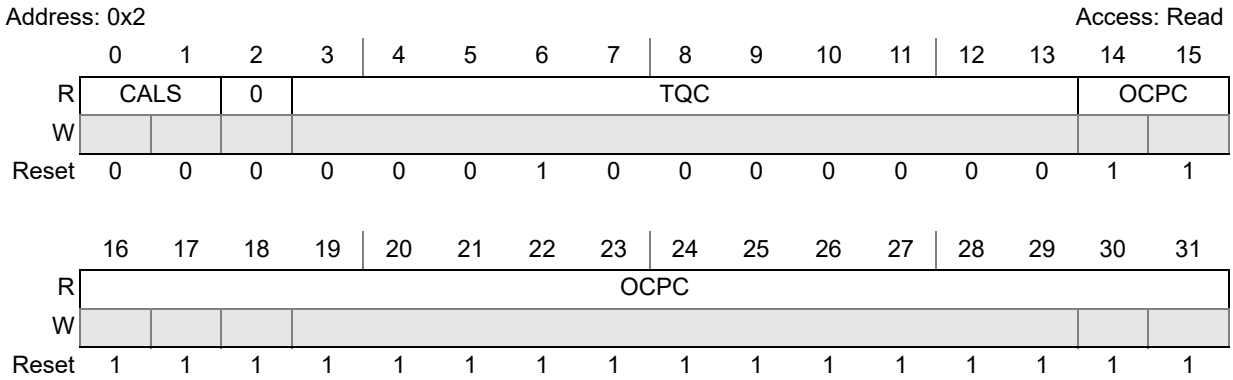


Figure 546. Calibration Status Register (CSTAT)

Table 601. Calibration Status Register field descriptions

Field	Description
0:1 CALS	Calibration State 00 Not_Calibrated 01 Basic_Calibrated 10 Precision_Calibrated 11 reserved
2	Reserved
3:13 TQC	0x000–0x3FF Time Quanta Counter Captured number of time quanta in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.
14:31 OCPM	0x00000–0x3FFFF Oscillator Clock Period Counter Captured number of oscillator clock periods in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

44.3.16.3.5 Calibration Watchdog Register (CWD)

The calibration watchdog is started after the first falling edge when the calibration FSM is in state Not_Calibrated (CSTAT.CALS = “00”). In this state the calibration watchdog monitors the message received input (**cu_mrx**). In case no message was received until the calibration watchdog has counted down to zero, the calibration FSM stays in state Not_Calibrated (CSTAT.CALS = “00”), the counter is reloaded with RWD.WDC and basic calibration is restarted after the next falling edge.

When in state Basic_Calibrated (CSTAT.CALS = “01”), the calibration watchdog is restarted with each received message (**cu_mrx** = ‘1’). In case no message was received until the calibration watchdog has counted down to zero, the calibration FSM returns to state Not_Calibrated (CSTAT.CALS = “00”), the counter is reloaded with RWD.WDC and basic calibration is restarted after the next falling edge.

When a quartz message was received, state Precision_Calibrated (CSTAT.CALS = “10”) is entered and the calibration watchdog is restarted. In this state the calibration watchdog monitors the quartz message received input (**cu_qmrx**). In case no message from a quartz controlled node was received by the attached M_CAN until the calibration watchdog has

counted down to zero, the calibration FSM transits back to state Basic_Calibrated (CSTAT.CALS = “01”).

The Calibration Watchdog Counter is clocked by the sample point input (**cu_spt**). The duration of a cu_spt pulse is one **cu_tqc** period. The signal is active when the CAN protocol engine on the attached M_CAN is started i.e. when the INIT bit is reset.

A calibration watchdog event is signaled by output **cu_cwe**. The duration of a **cu_cwe** pulse is one **cu_hclk** period.

A calibration watchdog event also sets interrupt flag **cuir.cwe**. If enabled by **cuie.cwee**, interrupt line cu_int is activated (set to high). Interrupt line cu_int remains active until interrupt flag **cuir.cwe** is reset.

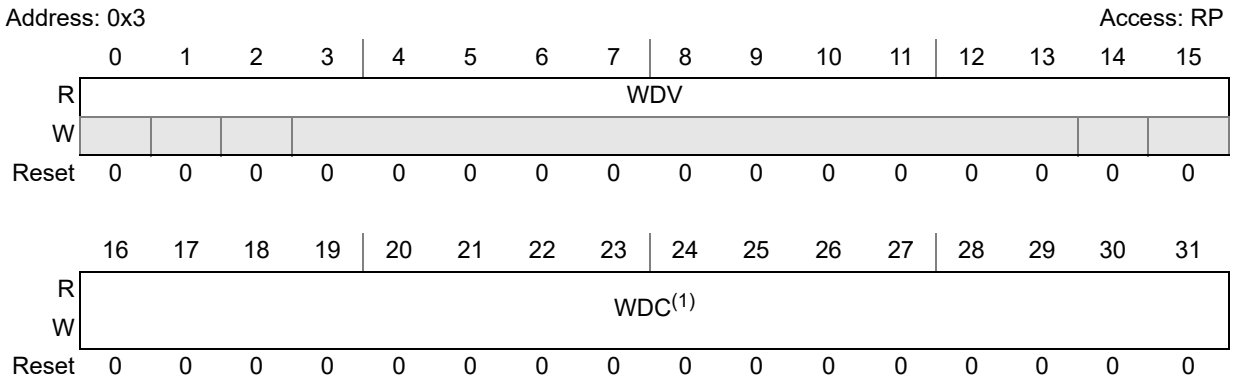


Figure 547. Calibration Watchdog (CWD)

1. Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when input cu_cce = ‘1’. Signal cu_cce is activated when the M_CAN control bits CCCR.CCE = ‘1’ AND CCCR.INIT = ‘1’.

Table 602. Calibration Status Register field descriptions

Field	Description
0:15 WDV]	Calibration State 00 Not_Calibrated 01 Basic_Calibrated 10 Precision_Calibrated 11 reserved
16:31 WDC	Reserved

44.3.16.3.6 CU Interrupt Register (CUIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. A hard reset will clear the register. The configuration of CUIE controls whether an interrupt is generated.

Address: 0x4 Access: W1C

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														CSC	CWE
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 548. CU Interrupt Register (CUIR)

Table 603. CU Interrupt Register field descriptions

Field	Description
30 CSC	Calibration State Changed 00 Calibration State unchanged 01 Calibration State has changed
31 CWE	Calibration Watchdog Event 00 No Calibration Watchdog Event 01 Calibration Watchdog Event occurred

44.3.16.3.7 CU Interrupt Enable Register (CUIE)

The settings in the CU Interrupt Enable register determine whether a status change in the CU Interrupt Register will be signaled on an interrupt line.

Address: 0x5 Access: RW

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														CSCE	CWEE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 549. CU Interrupt Enable Register (CUIE)

Table 604. CU Interrupt Enable Register field descriptions

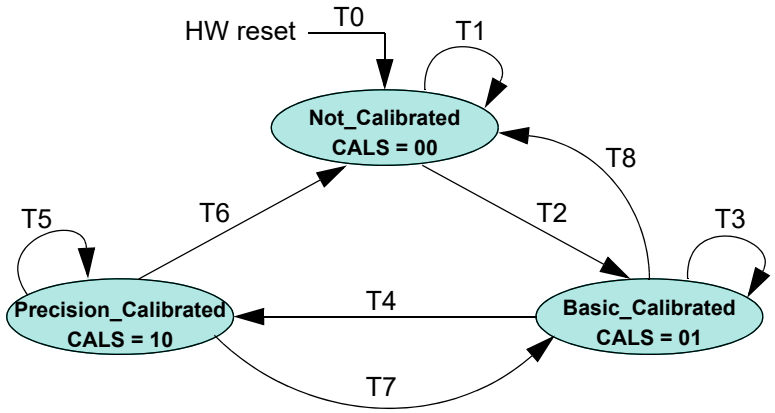
Field	Description
30 CSCE	Calibration State Changed Enable 0 Interrupt disabled 1 Interrupt enabled
31 CWEE	Calibration Watchdog Event Enable 0 Interrupt disabled 1 Interrupt enabled

44.3.16.4 Functional Description

Calibration of the time quanta clock **cu_tcq** via CAN messages is performed by adapting a clock divider that generates the CAN protocol time quantum tq from the clock **cu_pclk**.

- First step: Basic Calibration–The minimum distance between two edges from recessive to dominant is measured, this time to be assumed two CAN bit times, counted in PLL clock periods. The clock divider is updated each time a new measurement finds a smaller distance between edges. Basic calibration is achieved when the CAN protocol controller detects a valid CAN message. This is signaled to the Calibration on CAN unit by input signal **cu_mrx**.
- Second step: Precision Calibration – The calibration state machine measures the length of a longer bit sequence inside a CAN frame by counting the number of cu_pclk periods. The length of this bit sequence can be configured to 32 or 64 bits via CCFG.CLF. For a calibration field length of 32-/64- bit calibration message with at least 2/6 byte data field is required. The counter is controlled by input signals **cu_calf** and **cu_spt**. The counter is enabled when cu_calf is set and stops after the configured number of bits has been received. Received bits are signaled by the M_CAN by signal cu_spt. Precision calibration is based on the new clock divider value calculated from the measurement of the longer bit sequence.

Figure 550. Calibration FSM



T0: HW reset

T1: check for minimum

T2: message received (**cu_mrx** = '1')

T3: evaluate messages

T4: quartz message received (**cu_qmrx** = '1')

T5: evaluate quartz messages

T6: configuration change enabled (**cu_cce** = '1')

T7: watchdog event: no quartz message received

T8: watchdog event: no message received
or configuration change enabled (**cu_cce** = '1')

A change of the calibration state is signaled via by output **cu_csc**. The duration of a **cu_csc** pulse is one cu_hclk period.

A change of the calibration state also sets interrupt flag **cuir.csc**. If enabled by **cuie.csce**, interrupt line cu_int is activated (set to high). Interrupt line cu_int remains active until interrupt flag **cuir.csc** is reset.

Until precision calibration is achieved (cu_cok='1'), the attached M_CANs operate in a restricted mode (no frame transmission, no error or overload flag transmission, no error counting). In case calibration of the PLL is done by software by evaluating the calibration status from register CSTAT, the attached M_CANs have to be set to restricted operation mode (CCCR.ASM = '1') until the Calibration on CAN unit is in state Precision_Calibrated (see *Section 44.3.16.4.3: Software Calibration*).

Precision calibration may be performed only on valid CAN frames transmitted by a node with a stable, quartz-controlled clock. Calibration frames are detected by the M_CAN's acceptance filtering and are signaled to the Clock Calibration on CAN unit by signal cu_qmrx. A filter element and an Rx Buffer have to be configured in the attached M_CAN to identify and store calibration messages. After reception of a calibration message the Rx Buffer's new data flag has to be reset to enable signaling of the next calibration message.

In case there is only one CAN transmitter with a quartz clock in the network, this node has to transmit its first message after startup with at least one "1010" sequence in the data field or in the identifier. This assures that the non-quartz nodes can enter state Basic_Calibrated and then acknowledge the quartz node's messages.

Precision calibration must be repeated in predefined maximum intervals supervised by the calibration watchdog.

Note: When the Clock Calibration on CAN unit transits from state Precision_Calibrated back to Basic_Calibrated, the calibration OK signal is deasserted, the attached M_CANs complete ongoing transmissions, and then enter restricted operation (no frame transmission, no error or overload flag transmission, no error counting).

44.3.16.4.1 Configuration

The Clock Calibration on CAN Unit is configured via register CCFG. The register is only writeable while **cu_cce** ='1', i.e. when the attached M_CAN has CCCR.CCE and CCCR.INIT set.

For basic calibration the minimum number of oscillator periods between two consecutive falling edges at pin **cu_rx** is measured. The number of clock periods depends on the clock frequency applied at input **cu_pclk**. In case the measured number of clock periods is below the minimum configured by CCFG.OCPM e.g. due to a glitch on **cu_rx**, the value is discarded and measurement continues.

It is recommended to configure CCFG.OCPM slightly below two CAN bit times:

CCFG.OCPM < ((2 x CAN bit time) / cu_pclk period) / 32

The length of the bit field used for precision calibration can be configured to 32 or 64 bits via CCFG.CFL. The number of bits used for precision calibration has an impact on calibration accuracy and the maximum distance between two calibration messages. The reception of the bits is signaled by the attached M_CAN by input **cu_spt**.

The number of time quanta per bit time configured by CCFG.TQBT is used together with the measured number of oscillator clock periods CSTAT.OCPC to define the number of oscillator clocks per bit time.

When the clock calibration is bypassed by configuring CCFG.BCC = '1', the internal clock divider has to be configured via CCFG.CDIV to fulfill the condition m_can_cclk $\leq$ m_can_hclk.

Note: When clock calibration on CAN is active (CCFG.BCC = '0'), the baud rate prescaler of the attached M_CANs has to be configured to inactive.

44.3.16.4.2 Status signaling

The status of the Clock Calibration on CAN Unit can be monitored by reading register CSTAT.

When in state Precision_Calibrated the oscillator clock period counter CSTAT.OCPC signals the number of oscillator clock periods in the calibration field while CSTAT.TQC signals the number of time quanta in the calibration field.

The calibration state is monitored by CSTAT.CALS. A change of the calibration state is signaled at the module interface by cu_csc. The duration of a cu_csc pulse is one cu_hclk period. A change of the calibration state also sets interrupt flag CUIR.CSC. If enabled by CUIE.CSCE, interrupt line cu_int is activated (set to high). Interrupt line cu_int remains active until interrupt flag CUIR.CSC is reset.

A calibration watchdog event is signaled by output cu_cwe. The duration of a cu_cwe pulse is one cu_hclk period. A calibration watchdog event also sets interrupt flag CUIR.CWE. If

enabled by CUIE.CWEE, interrupt line cu_int is activated (set to high). Interrupt line cu_int remains active until interrupt flag CUIR.CWE is reset

44.3.16.4.3 Software Calibration

The Clock Calibration on CAN unit also supports software calibration of the clock generated from an adjustable PLL, connected to the clock input cu_pclk. For this purpose the clock from cu_pclk has to be routed to output cu_tqc by programming CCFG.BCC = '1'. The Clock Calibration on CAN unit's internal clock divider has to be configured via CCFG.CDIV to meet the clock requirements of the attached M_CANs. All other configuration parameters have to be set via CCFG as if the Clock Calibration on CAN unit is generating cu_tqc.

For startup the M_CANs attached to the Clock Calibration on CAN unit have to be configured for Restricted Operation (CCCR.ASM = '1'). The PLL clock has to be adjusted until the Clock Calibration on CAN unit has reached state Precision_Calibrated. Now the software has to reset CCCR.ASM and the M_CANs can start normal operation.

During operation the software has to check regularly whether the Clock Calibration on CAN unit is still in state Precision_Calibrated. In case the Clock Calibration on CAN unit has left state Precision_Calibrated due to drift of the PLL clock, the attached M_CANs have to be set into Restricted Operation Mode by programming CCCR.INIT, CCCR.CCE, and CCCR.ASM to '1'.

After the PLL clock has been adjusted successfully (Clock Calibration on CAN unit is in state Precision_Calibrated), the M_CANs can be set to normal operation again.

44.3.16.4.4 Signal description

Table 605 below lists the interface signals of the Calibration on CAN unit.

Table 605. Calibration on CAN Module Interface

Port	Direction	Description
Clocks and Reset		
cu_pclk	IN	clock input from clock select multiplexer
cu_hclk	IN	host clock
cu_reset	IN	module reset
Physical Layer Interface		
cu_rx	IN	CAN receive input
M_CAN Interface		
cu_cce	IN	configuration change enable
cu_spt	IN	sample point
cu_mrx	IN	message received
cu_qmrx	IN	quartz message received, to be connected to one of the M_CAN's filter event outputs
cu_calf	IN	calibration field
cu_aff	IN	acceptance filtering finished
cu_tqc	OUT	time quanta clock

Table 605. Calibration on CAN Module Interface(Continued)

Port	Direction	Description
cu_cok	OUT	calibration OK
cu_csc	OUT	calibration state changed
cu_cwe	OUT	calibration watchdog event
cu_int	OUT	CU interrupt

44.4 CAN RAM arbiter

This block acts as a dynamic arbiter between the various CAN nodes and the CPU for granting access to the shared CAN memory. The dynamic arbitration ensures:

- 50% bandwidth for CPU, 50% shared by the CAN nodes when simultaneous access requested from all masters and CPU,
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access.

Note: From the arbiter prospective the CPU bandwidth may be limited to 50%. The CPU may be waiting stated due to this bandwidth limitation when accessing the shared memory.

44.4.1 Features

- Active low asynchronous reset
- 1–CPU + 5–CAN arbiter.
- Combined dynamic arbitration scheme: Bandwidth between CPU and M_CAN interfaces can be redistributed. If active, the CPU interface is granted a 50% portion of the available bandwidth.
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access
- The SRAM access by a single master needs two cycles (address and data) for each read/write command. The arbiter has a pseudo address pre-fetching mechanism. This mechanism acts when multiple masters are accessing the SRAM and it overlaps the address cycle of one master with the data cycle of another master. In case of multiple masters accessing the SRAM, the pseudo address pre-fetching scheme saves multiple clock cycles.
- The peripheral GSI module enable acts as a request from each master. Each master will be made to wait at least one clock cycle before the grant is given. The arbiter ensures that the CPU does not wait for more than 1 clock cycle, hence ensuring a 50% guaranteed bandwidth.
- Time slot is defined as one peripheral clock cycle.

44.4.2 Functional overview using examples

The dynamic arbitration scheme is explained using the following examples.

- Example 1 (active CPU IF, 3 out of 3 CAN nodes are active):
  - CPU obtains every 2nd slot
  - CAN 1/2/3 obtains every 6th time slot
- Example 2 (inactive CPU IF, 4 out of 4 CAN nodes are active):
  - CPU obtains no time slots
  - CAN 1/2/3/4 obtains every 4th time slot
- Example 3 (active CPU IF, 2 out of 4 CAN node are active):
  - CPU obtains every 2nd time slot
  - Active CAN 1/2 obtains every 4th time slot
- Example 4 (inactive CPU IF, 2 out of 4 CAN node are active):
  - CPU obtains no time slots
  - Active CAN 1/2 obtains every 2nd time slot

44.5 SRAM interface and memory organization

The CAN subsystem will interface with an external RAM using this interface. The RAM sizes required for the CAN subsystem are summarized in the following sections.

44.5.1 Shared memory

Table 606 shows maximum possible configuration per Filter, FIFO and Buffer but not all of these items can be configured to maximum size at the same time due to the smaller implemented physical memory as listed in *Table 607: Shared memory map vs CAN transmit, receive and filters elements*.

Table 606. M_CAN message memory size

Memory element	Number of elements	32-bit word per element	Total 32-bit words
11-bit Filter	128	1	128
29-bit Filter	64	2	128
Rx FIFO 0	64	4	256
Rx FIFO 1	64	4	256
Rx Buffer	64	4	256
Tx Event FIFO	32	2	64
Tx Buffers	32	4	128
M_CAN message memory size (word)			1216

Refer *Table 607: Shared memory map vs CAN transmit, receive and filters elements* for the available physical memory space used in the device.

44.5.2 ECC controller

The RAM embeds ECC logic and code to provide Single Error Correction / Double Error Detection (SECCDED). The module does not guarantee any proper functionality if more than 2 bits are in errors. It supports only 32-bit write access and 8-/16-/32-bit read access. The ECC error is reported to each CAN module as well as available at the CAN subsystem top, so that the same can be forwarded to the Error Management Module.

The module uses Hamming code for single error correction and double error detection logic. It involves transmitting data with multiple ecc_check bits and decoding the associated ecc_check bits when receiving data to detect errors. The SECDED Hamming code is not able to detect three bit errors. Rather, in the presence of a three bit error, a conventional SECDED code returns an error code that is indistinguishable from an error code resulting from a single bit error. Hence, this module does not guarantee any proper functionality when more than 2 bits are in error.

The ecc_check bits are parallel parity bits generated from XORing certain bits in the original data word. If bit error(s) are introduced in the codeword, several ecc_check bits show parity errors after decoding the retrieved codeword. The combination of these ecc_check bit errors display the nature of the error. In addition, the position of any single bit error is identified from the ecc_check bits.

The ECC error address is reported at the top of the subsystem using a 16-bit ECC error address signal. This address is valid only when the ECC bit error output is valid.

44.5.2.1 Features

The ECC module supports the following:

- ecc_error = 00 ' No error is detected in the read data from memory
- ecc_error = 01 ' This indicates single bit error occurred within the 39-bit codeword. In addition, the error is corrected, and the forwarded data to the master is error free.
- ecc_error = 10 ' This indicates that a two bit error has occurred within the codeword. In this case, no error correction is possible.
- ecc_error = 11 ' This indicates that errors beyond the detection capability have occurred within the codeword and no error correction is possible. This is an invalid error type.

44.5.3 Transmit message buffer locking mechanism

Note: Refer to Device Configuration chapter to see the number of CAN nodes used in the device.

This block will implement a transmit message buffer locking mechanism using the CAN nodes internal flags output.

The Host controller has to access the Message RAM for updating transmit message buffers (e.g. the data section), the M_CAN controller needs access to the Message RAM to execute acceptance and transmit filtering in time and to prepare the next message for transmission by transferring it to the CAN core.

To avoid message buffer corruption, further host accesses to message buffers which are requested for transmission should be prevented. Otherwise, if the Host updates a message buffer which is simultaneously transferred to the CAN core, inconsistent data may be transmitted on the CAN bus.

This functionality is achieved using the “transmit buffer request pending” (M_CAN) information of the CAN modules. With the information from these signals, the message buffer locker prevents Host accesses from transmitting message buffers which are requested and pending for transmission.

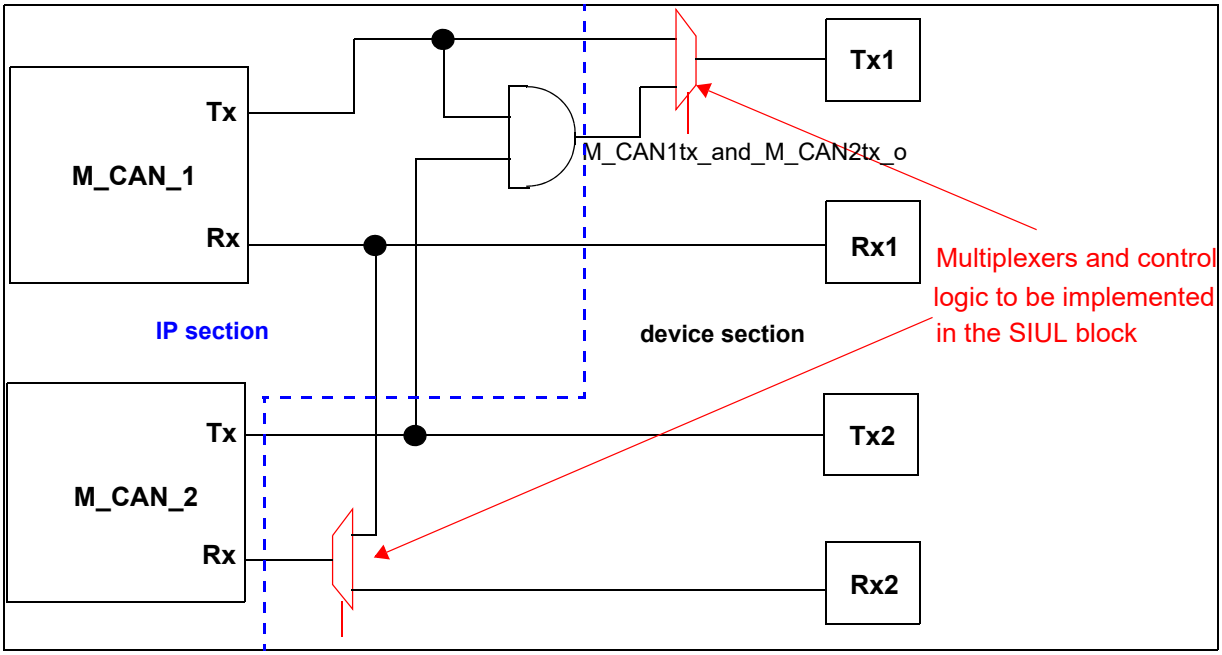
44.5.3.1 Transmit message buffer access locking for M-CAN

Each CAN module can have at most 32 TX Buffers and the size of each TX Buffer is 4 words. The CPU write locking mechanism ensures that whenever any TX Buffer's pending request is set (by the TRP bit of the TXBRP register), the CPU does not get any write access to the corresponding TX buffers. If the CPU tries to write into a TX Buffer whose pending request is set it receives an error response on the CAN RAM Transfer Error port.

44.6 CAN nodes 1 and 2 I/Os sharing

Each CAN node has its Tx/Rx signals connected to separated GPIO pairs. It is required for M_CAN_1 and M_CAN_2 to share one common GPIO pair. This feature is described below.

Figure 551. M_CAN_1 and M_CAN_2 connections to GP I/Os



44.7 External signal description

The CAN subsystem has following external pins.

- M_CAN_x_RX: M_CAN x receive input
- M_CAN_x_TX: M_CAN x transmit output

44.8 Memory map and register description

44.8.1 Shared memory map

Table 607 shows the shared memory map vs CAN subsystem transmit, receive and filters elements.

Table 607. Shared memory map vs CAN transmit, receive and filters elements

Start Address Offset (byte)	End Address Offset (byte)	CAN block/sub block		Comment
0x0000	0x06DF	Reserved		
0x06E0	0x0CFF	M_CAN1	Standard Filters	This available physical memory space could be configured for Filters/ FIFO/Buffer according to the *Table 606*.
			Extended Filters	
			Rx FIFO 0	
			Rx FIFO 1	
			Rx Buffer	
			Tx Event FIFO	
0x0D00	0x0EFF		Tx Buffers	Protected from CPU write access using the M_CAN1 TXBRP register output
0x0F00	0x187F		Reserved	
0x1880	0x1BFF	M_CAN2	Standard Filters	This available physical memory space could be configured for Filters/ FIFO/Buffer according to the *Table 606*.
			Extended Filters	
			Rx FIFO 0	
			Rx FIFO 1	
			Rx Buffer	
			Tx Event FIFO	
0x1C00	0x1CFF		Tx Buffers	Protected from CPU write access using the M_CAN2 TXBRP register output
0x1D00	0x1DFF	Reserved		

Note: There is no write access protection from a CAN module to the message memory space. The write access protection is only for CPU access to the message memory.

Note: The start address for every memory section is to be configured by the user in the relevant CAN modules registers. However there is no hardware check in the RAM interface for the start address consistency.

Note: However the number of filters, Tx/Rx buffers, triggers used, it is mandatory for the user to configure the start address registers, specified in Table 608, in order to fit the memory mapping specified in Table 607.

Note: The CPU uses the peripheral interface to access the memory locations. It must be noted that there is no transfer error reported to the CPU in case of access to reserved memory locations.

Table 608. Detailed mapping in CAN subblocks

Address offset (byte)	CAN subblock	Element name	Comment
0x0000 --- 0x01FF	Standard Filters	Standard Message ID filter 0 --- Standard Message ID filter 127	address configured in SIDFC [FLSSA]
0x0200 --- 0x03FF	Extended Filters	Extended Message ID filter 0 --- Extended Message ID filter 63	address configured in XIDFC [FLESA]
0x0400 --- 0x07FF	Rx FIFO0	Rx FIFO0 element 0 --- Rx FIFO0 element 63	address configured in RXF0C[F0SA]
0x0800 --- 0x0BF3	Rx FIFO1	Rx FIFO1 element 0 --- Rx FIFO1 element 63	address configured in RXF1C[F1SA]
0x0BF4 ... 0x0BFF	Rx Buffer	Rx Buffer element 0 --- Rx Buffer element 63	address configured in RXBC[RBSA]
0x0C00 --- 0x0CFF	Tx Event FIFO	Tx event FIFO element 0 --- Tx event FIFO element 31	address configured in TXEFC[EFSA]
0x0D00 --- 0x0EFF	Tx Buffers	Tx Buffer element 0 --- Tx Buffer element 31	address configured in TXBC[TBSA]

45 Serial Interprocessor Interface (SIPI)

45.1 Introduction

The Serial Interprocessor Interface (SIPI) is an application layer protocol that runs on top of the LFAST module. It is used by the local device to access the shared memory of a remote device. SIPI defines point-to-point full duplex communication between two devices, where LFAST works as a physical medium of communication between both the devices.

45.1.1 Scalability

The SIPI protocol is designed to provide a sophisticated, high bandwidth, multimaster, multi-channel memory interface between two devices using few interconnecting signals. But the protocol is designed in such a way that a subset of the protocol can be implemented, where die area is more important than features.

Main scalable features:

- Number of concurrent channels:
  - Full Implementation = 4
  - Minimum Implementation = 1
- Full implementation has a node both as Initiator and Target of commands. Minimum Implementation either Initiator or Target
- Full Implementation includes a block transfer feature, but this feature is optional.

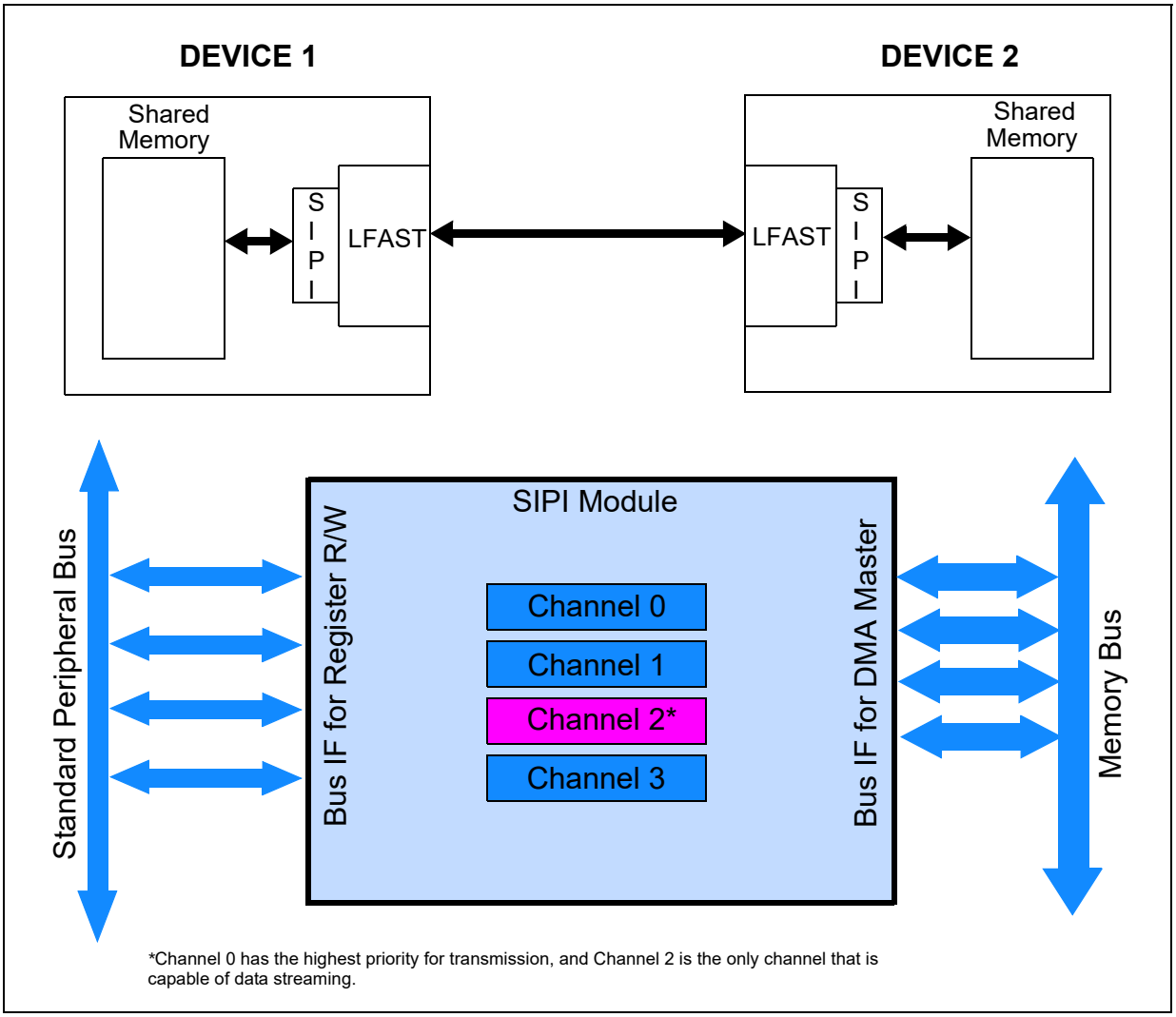
The rest of this section describes a full implementation of SIPI, which includes:

- Advanced High-Performance Bus (AHB-Lite) master interface
- Direct Memory Access (DMA) interface
- LFAST Tx/Rx (transmit/receive) interfaces along with Peripheral Bus Interface (IPS).

45.2 Overview

An instance of SIPI can act as initiator, or target or both. SIPI can access the shared memory directly through its AHB master interface or through its DMA interface. DMA interface is used when the node acts as an initiator while the AHB Master interface will be used when the node acts as target. SIPI has four channels, with one channel (Channel 2) having data streaming capability. Payload width for channel 0, 1 and 3 is 32 bits, and for channel 2 data widths can be 32 bits, or 256 bits when streaming. Any of these channels can be used for DMA access or bus interface access depending on the setting of the SIPI_CCR*n*. CRC encoder calculates the CRC on the command frame. Then the SIPI initiator appends the CRC to the end of the frame before transmission.

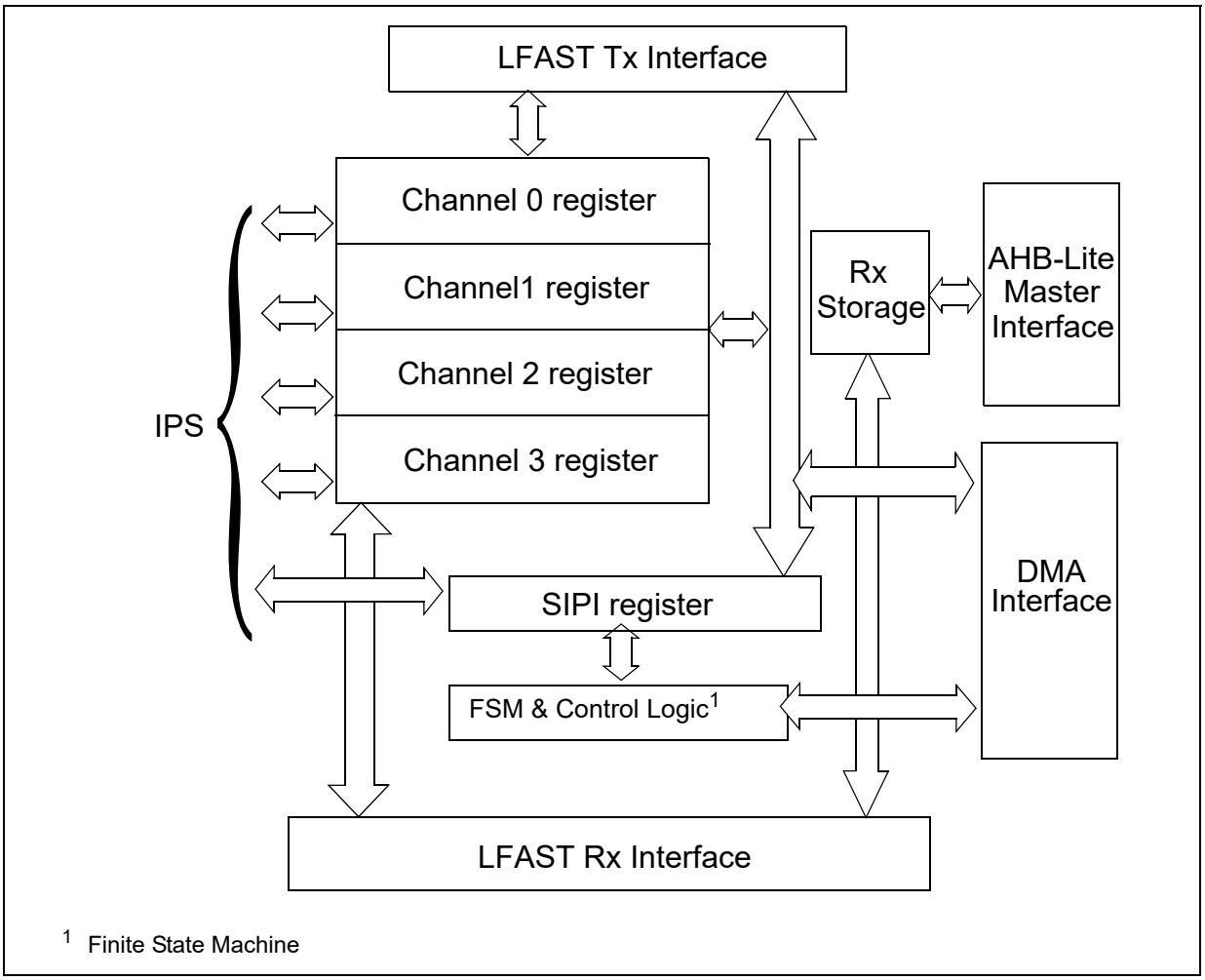
Figure 552. Interprocessor communication diagram



*Channel 0 has the highest priority for transmission, and Channel 2 is the only channel that is capable of data streaming.

45.3 SIPI block diagram

Figure 553. SIPI block diagram



45.4 Feature description

This section describes the features of the SIPI module.

45.4.1 Main features

- Point-to-point communication between two devices
- Full duplex communication
- Four channels, including one channel with data streaming capability
- Configurable DMA access for each channel
- CRC protection mechanism
- Timeout protection mechanism
- Fixed priority channel selection
- Data size up to 256 bits on streaming channel
- Common tag pool for assigning sequential transfer IDs to every new transfer
- AHB master interface that is used by target node to access shared memory
- Target node contains a set of nine 32-bit internal registers to store commands
- Up to two outstanding requests are supported at initiator

45.4.2 Standard features

- IPS bus interface
- SPP DMA2x bus interface
- AHB Master Interface
- LFAST Tx/Rx interfaces
- Cyclic Redundancy Check error detection (CRC16)

45.5 SIPI operation from reset

When the SIPI module exits reset, it is operational and target mode is enabled without the need to configure the control registers.

45.6 Functional description

45.6.1 External signals

The SIPI has no chip external signals.

45.6.2 Frame format

All frames have the same general format:

- 16-bit header
- Address, Address and Data, or nothing
- 16-bit CRC

There are 2 main groups of command; read and write. Within those 2 groups are three read/write formats:

- 32-bit
- 16-bit
- 8-bit

Each command generates one of three different responses:

- Read data
- Write acknowledge
- Error

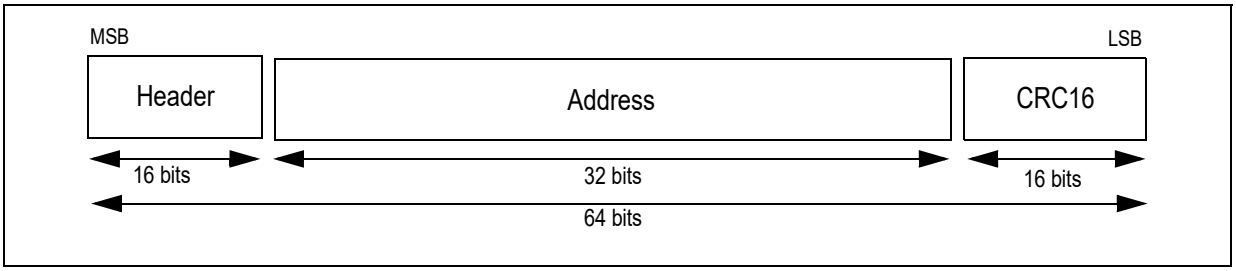
There is one additional command that requests the module ID from the Target node. The ID is a unique 32-bit ID that is specific to a particular device. It is normally the same as the JTAG ID.

The sections below illustrate the four different frame formats that are used to implement all the commands and responses.

The number of frames depends on the data buffers (associated with every channel), the frame data is stored in data buffers at the initiator. Address and data are both transmitted in the same order in which they are stored.

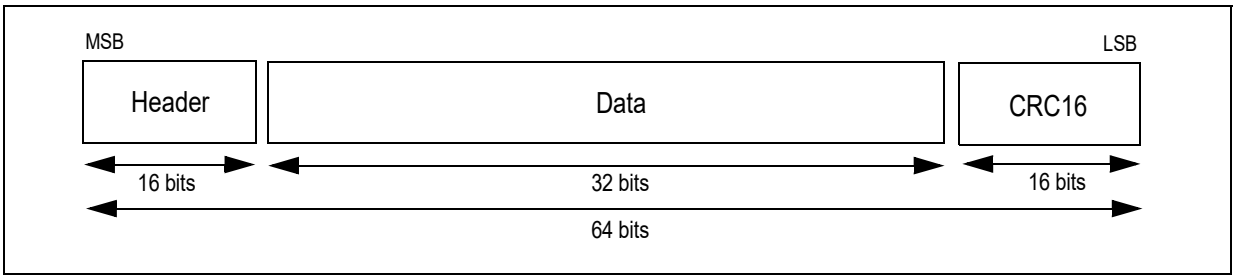
45.6.2.1 Register read request

Figure 554. SIPI register read request



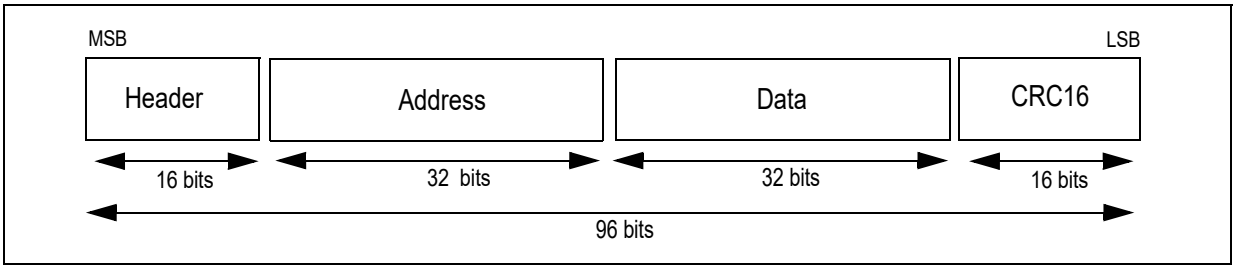
45.6.2.2 Register read response, ID request response

Figure 555. SIPI read response



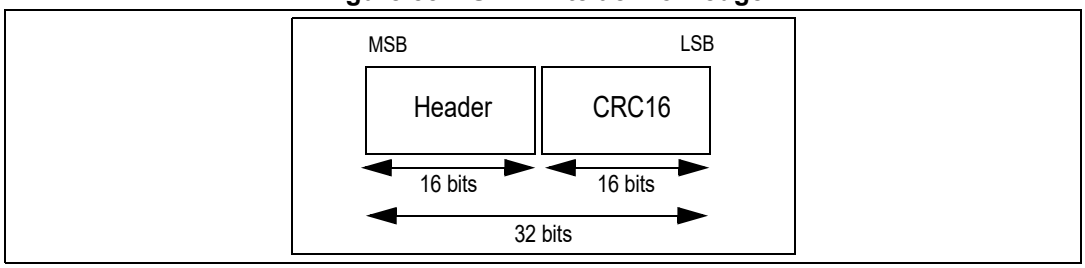
45.6.2.3 Register write request

Figure 556. SIPI register write request



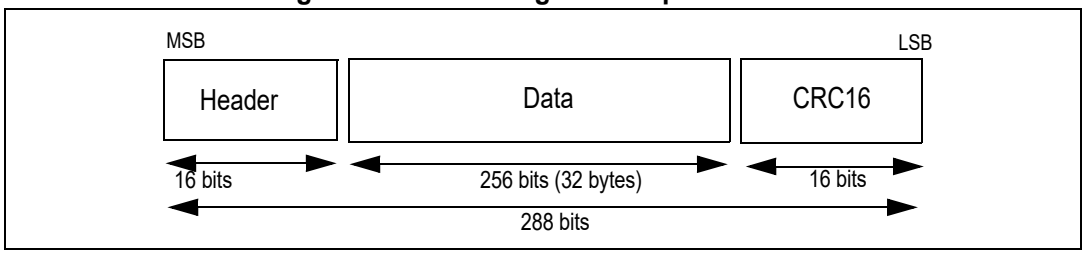
45.6.2.4 Trigger transfer, ID transfer, write acknowledge and streaming write acknowledge

Figure 557. SIPI write acknowledge



45.6.2.5 Streaming write request

Figure 558. Streaming write request format



Note: Direct write operations are used to set the streaming address.

Streaming data write is performed using the format shown in Figure 558.

45.6.2.6 Header field

Header field contains 16 bits of configuration information. MSB will be transmitted first.

45.6.2.6.1 SIPI header coding

Figure 559, *Table 609*, *Figure 560*, and *Table 610* show how the SIPI header bits are coded.

Figure 559. SIPI header coding

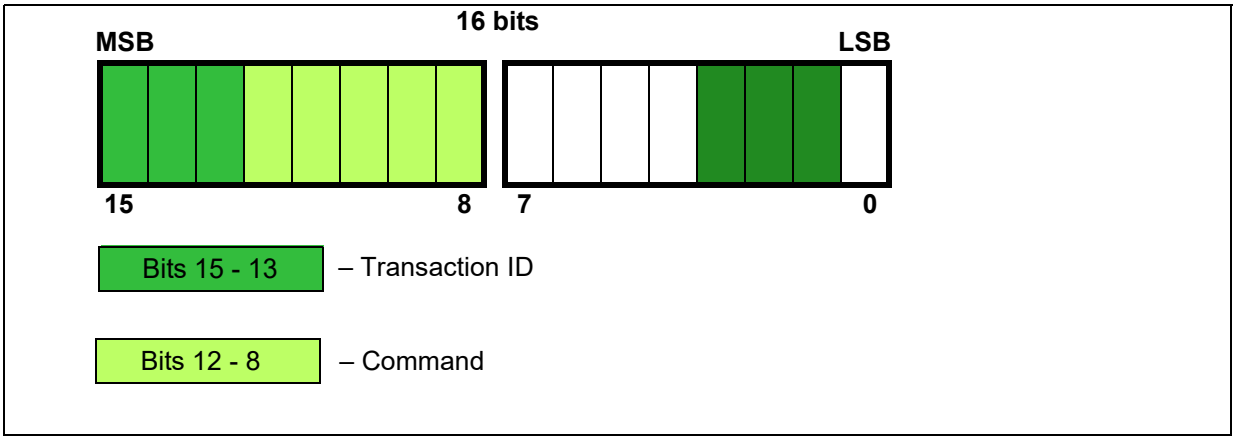


Table 609 shows the command coding for the bits[12:8] of the header.

Table 609. SIPI header command coding

b[12:8] (hex)	Command	Payload Size
00	Read 8 bits	64
01	Read 16 bits	64
02	Read 32 Bits	64
03	Reserved	—
04	Write 8 bits with ACK	96
05	Write 16 bits with ACK	96
06	Write 32 bits with ACK	96
07	Reserved	—
08	ACK – OK	32
09	ACK – Fault	32
0A	Read Answer – OK	64
0B	Reserved	—
0C	Trigger comm and with ACK	32
0D	Reserved	—
…	…	…
11	Reserved	—
12	ID Register Read Request	32
13	Reserved	—
…	…	…
16	Reserved	—
17	Stream Data with ACK (32 bytes)	288
18	Reserved	—

Table 609. SIPI header command coding(Continued)

b[12:8] (hex)	Command	Payload Size
…	…	...
1F	Reserved	—

Figure 560 shows the contents of bits[7:0] in the SIPI header.

Figure 560. SIPI header channel field

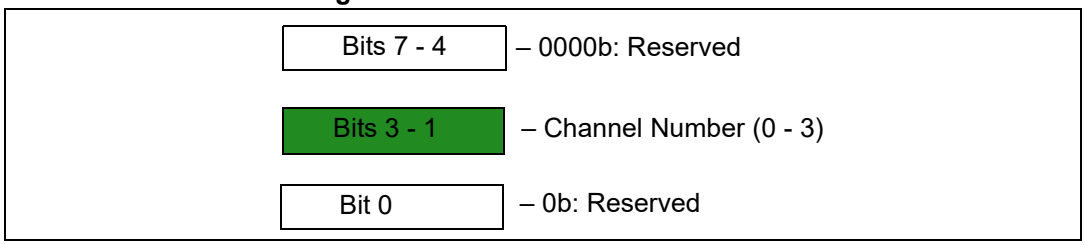


Table 610 shows the channel number coding.

Table 610. SIPI Header Channel Number Coding

SIPI Channel Name	Channel Number Coding in header(s)		Comment
	SIPI Channel Coding Select Bit		
	Code Table I (1)	Code Table II (0)	
0 (Channel A)	000b	100b	In use
1 (Channel B)	001b	101b	In use
2 (Channel C)	010b	110b	In use
3 (Channel D)	011b	111b	In use
4 (Channel E)	100b	000b	Reserved
5 (Channel F)	101b	001b	Reserved
6 (Channel G)	110b	010b	Reserved
7 (Channel H)	111b	011b	Reserved

45.6.2.7 Address field

The address field is 32 bits wide with the MSB transmitted first.

45.6.2.8 Payload field

Table 611 shows the payload sizes of LFAST frames.

Table 611. Payload size of LFAST channel frame

LFAST Code	SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
000b	—	8	1
001b	—	32	4
010b	010b	64	8
011b	011b	96	12
100b	100b	128	16
101b	101b	256	32
110b	110b	512	64
111b	111b	288	36

Table 612 shows the converted coding of LFAST for a given SIPI code.

Table 612. Converted coding of LFAST channel code for SIPI headers

LFAST Code	SIPI Code	Channel (hex)(1)
0100b	100b	A
0101b	101b	B
0110b	110b	C
0111b	111b	D
1000b	000b	E (not used by SIPI)
1001b	001b	F (not used by SIPI)
1010b	010b	G (not used by SIPI)
1011b	011b	H (not used by SIPI)

1. SIPI channel 0 sends all commands on LFAST channel A, commands received on LFAST channel A, are processed and the response sent back on LFAST channel A.
SIPI channel 1 sends all commands on LFAST channel B, commands received on LFAST channel B, are processed and the response sent back on LFAST channel B.
SIPI channel 2 sends all commands on LFAST channel C, commands received on LFAST channel C, are processed and the response sent back on LFAST channel C.
SIPI channel 3 sends all commands on LFAST channel D, commands received on LFAST channel D, are processed and the response sent back on LFAST channel D.

45.6.2.9 CRC field

CRC field is 16 bits wide with calculation always enabled using CRC-16-CCITT syndrome (x16 + x12 + x5 + 1). The data stream is transmitted with the MSB sent first.

45.6.2.9.1 CRC examples

45.6.2.9.1.1 Example 1 – 32-bit write on channel 1 with ID1

- Header = 260Ah
- Address = 1122_3344h
- Data = CCDD_EEFFh
- CRC = BF7Dh

45.6.2.9.1.2 Example 2 – 32-bit read on channel 2 with ID2

- Header = 420Ch
- Address = 89AB_CDEFh
- CRC = 6B80h

45.6.2.9.1.3 Example 3 – Event command on channel 3 with ID3

- Header = 6C0Eh
- CRC = B286h

45.7 Transfer types

This section describes the available transfer types of the SIPI module. The SIPI frame is inserted inside the payload of the LFAST frame as shown in the figures of the following examples.

45.7.1 Read transfer

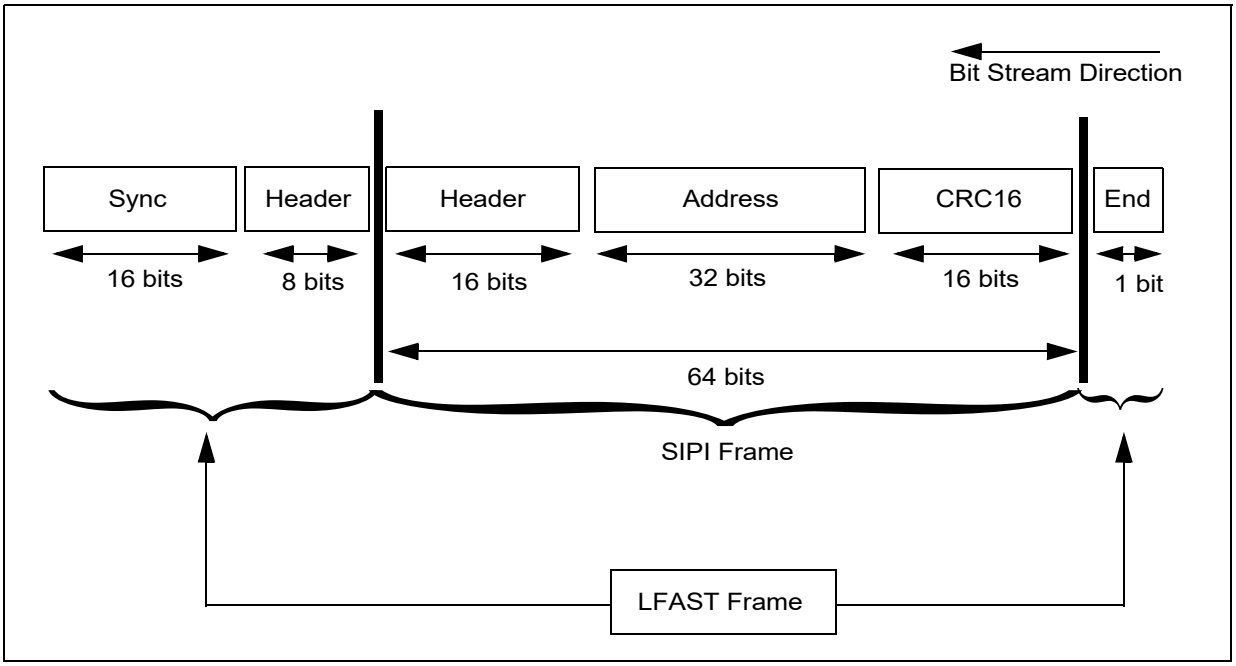
A Read transfer can be of two types:

- Read request transfer (at initiator node)
- Read response transfer (at target node)

45.7.1.1 Register read request transfer

If there is a read request transfer, the initiator node will send header, address and CRC bits as shown in *Figure 561*.

Figure 561. Read request transfer (showing LFAST frame encapsulation)



45.7.2 Register read answer transfer

In response to a read request by the initiator node, the target node will send header, payload and CRC16. Data transfer could be in 8-bit, 16-bit or 32-bit modes (see *Figure 562*). In the case of 8-bit or 16-bit modes, copies of the SIPI data is sent in the payload (see *Figure 563*).

Figure 562. Read answer transfer (LFAST frame encapsulation is not shown)

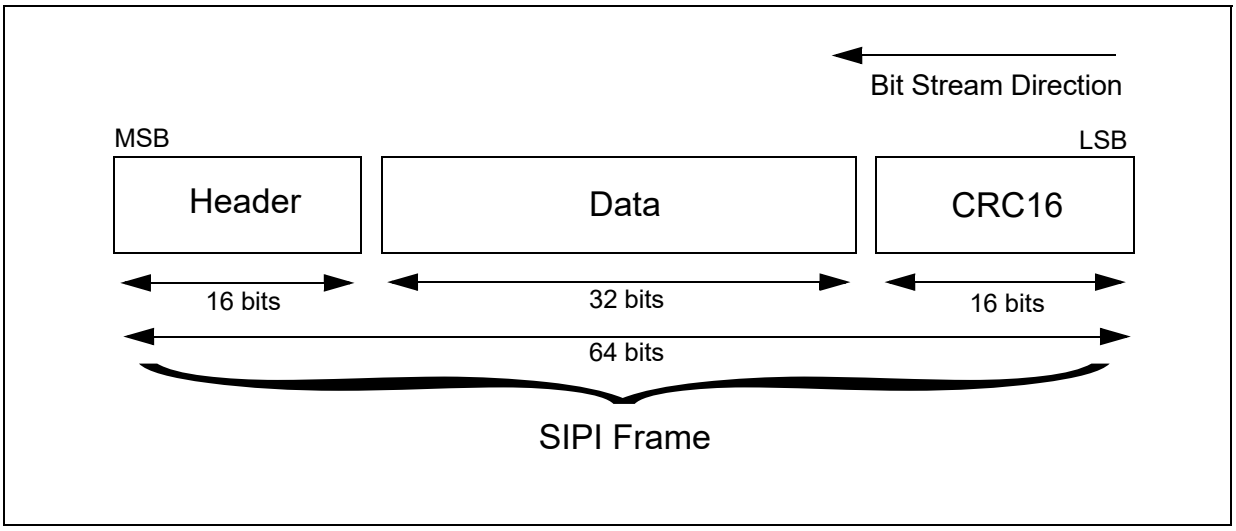
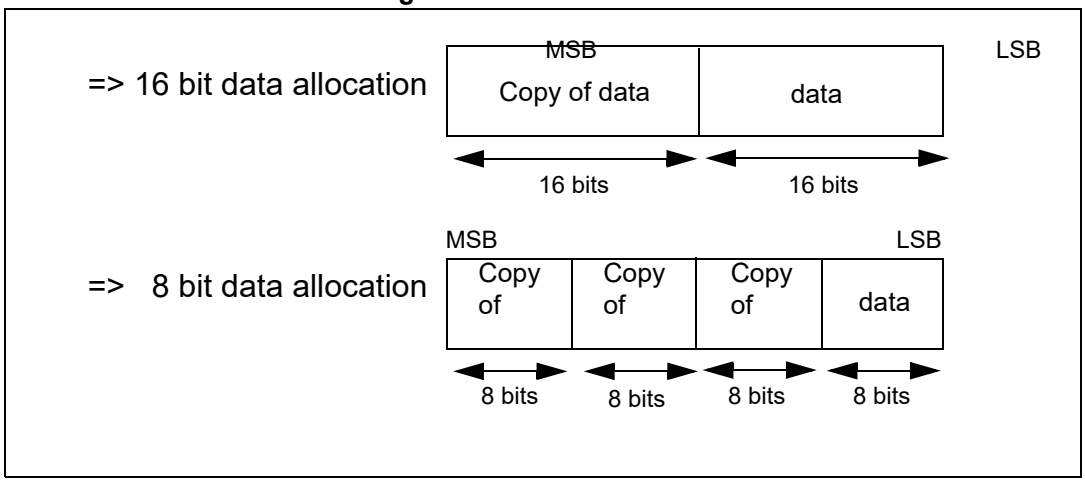


Figure 563. Data allocation



Note: For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.

- *Bits[1:0]:*
  - *00 - byte 3 enabled (MSB)*
  - *01 - byte 2 enabled*
  - *10 - byte 1 enabled*
  - *11 - byte 0 enabled (LSB)*

For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as half word enable.

- Bit[0]:
  - 0 - half-word 1 enabled (MSB)
  - 1- half-word 0 enabled (LSB)

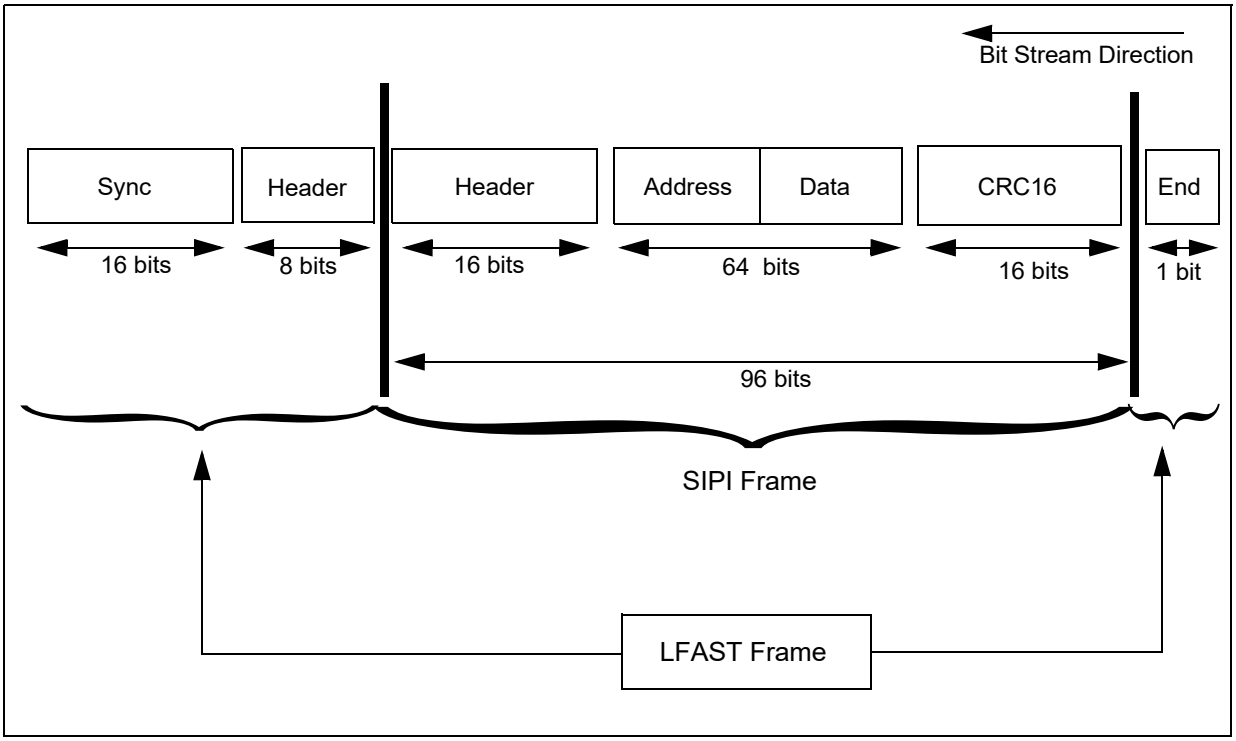
45.7.3 Register Write transfer

Register Write transfer can be of two types:

- Normal Write transfer – channels 0, 1, 2 and 3
- Streaming Data transfer – channel 2 only

A Register Write transfer can be done through Normal Write transfer - channels 0, 1, 2 and 3 (see *Section 45.7.3.1: Normal write transfer*) as shown in *Figure 564*.

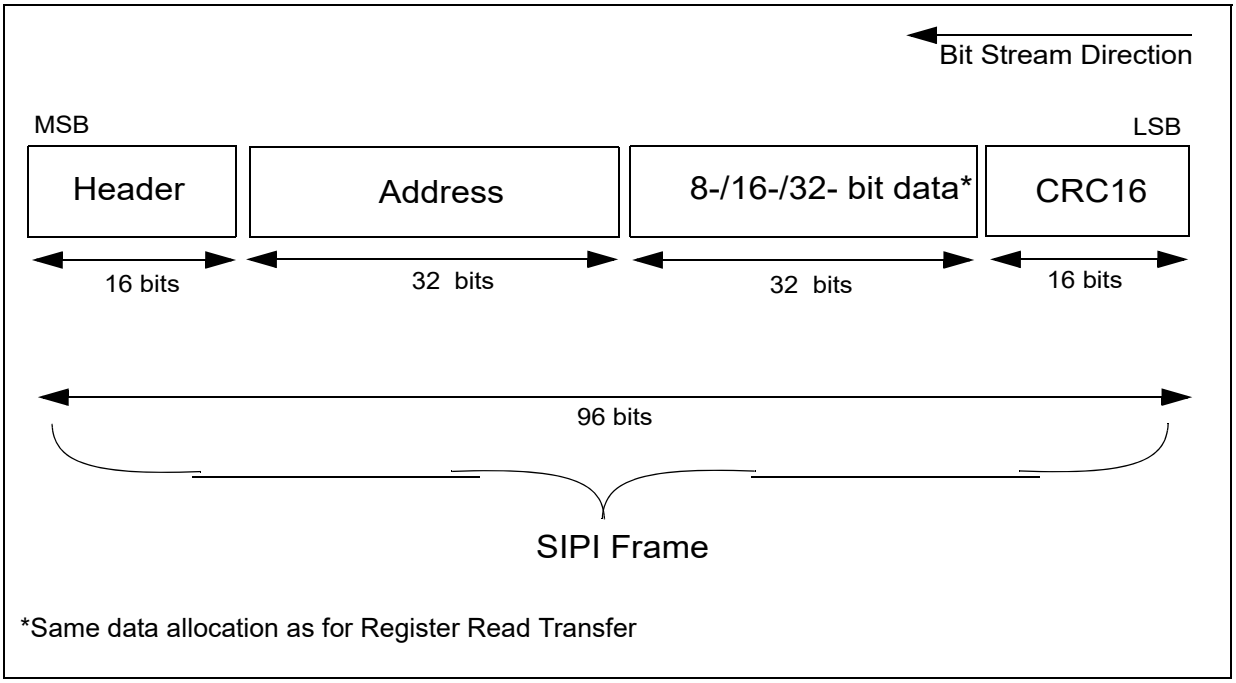
Figure 564. Register write transfer (showing LFAST frame encapsulation)



45.7.3.1 Normal write transfer

A normal write transfer contains header, address, data (32-bit) and 16-bit CRC as shown in *Figure 565*.

Figure 565. Write transfer (LFAST frame encapsulation is not shown)



Note: In the case of an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as a byte enable.

- *Bits[1:0]:*
  - *00 – byte 3 enabled (MSB)*
  - *01 – byte 2 enabled*
  - *10 – byte 1 enabled*
  - *11 – byte 0 enabled (LSB)*

In the case of a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as a half-word enable.

- *Bit[0]:*
  - *0 – half-word 1 enabled (MSB)*
  - *1 – half-word 0 enabled (LSB)*

45.7.3.2 Streaming write transfer

Streaming write transfers have 256 bits of payload.

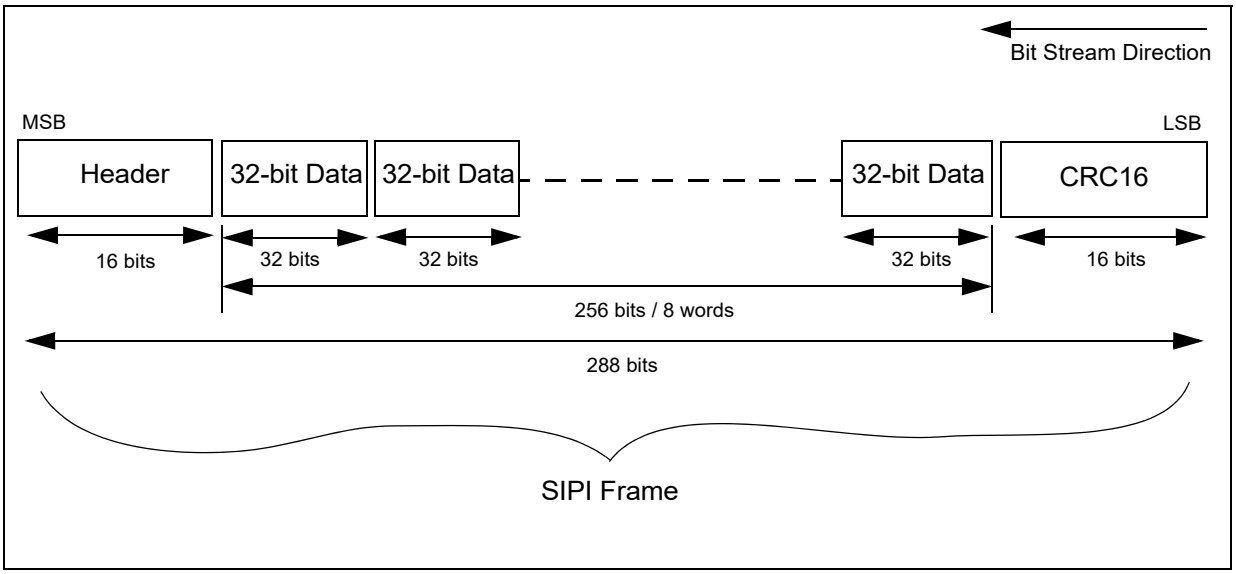
45.7.3.2.1 Streaming write transfer with address

Setting the address is performed using a direct write transfer. SIPI_MAXCR and SIPI_ARR are written before the SIPI_ACR. This is to ensure that the unwanted behavior of the SIPI trying to access non-shared memory is avoided. This is only true for the very first streaming command, subsequent streaming command(s) may not have the need to write SIPI_MAXCR and SIPI_ARR (see *Section 45.15.1.10: SIPI Max Count Register (SIPI_MAXCR)*, *Section 45.15.1.12: SIPI Address Count Register (SIPI_ACR)*, and *Section 45.15.1.11: SIPI Address Reload Register (SIPI_ARR)*.

45.7.3.2.2 Streaming transfer with data

Figure 566 shows the packet structure of streaming transfers that contain data.

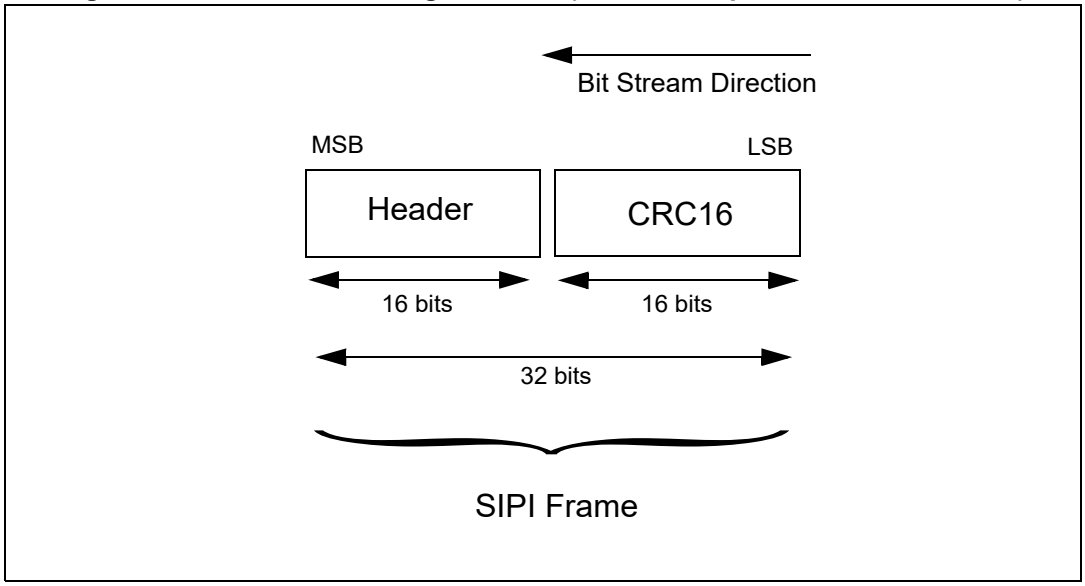
Figure 566. Streaming transfer with data



45.7.4 Write Acknowledge transfer

A Write Acknowledge transfer contains only header and CRC bits (see *Figure 567*). The CRC bits are calculated on the header field.

Figure 567. Write Acknowledge transfer (LFAST encapsulation is not shown)

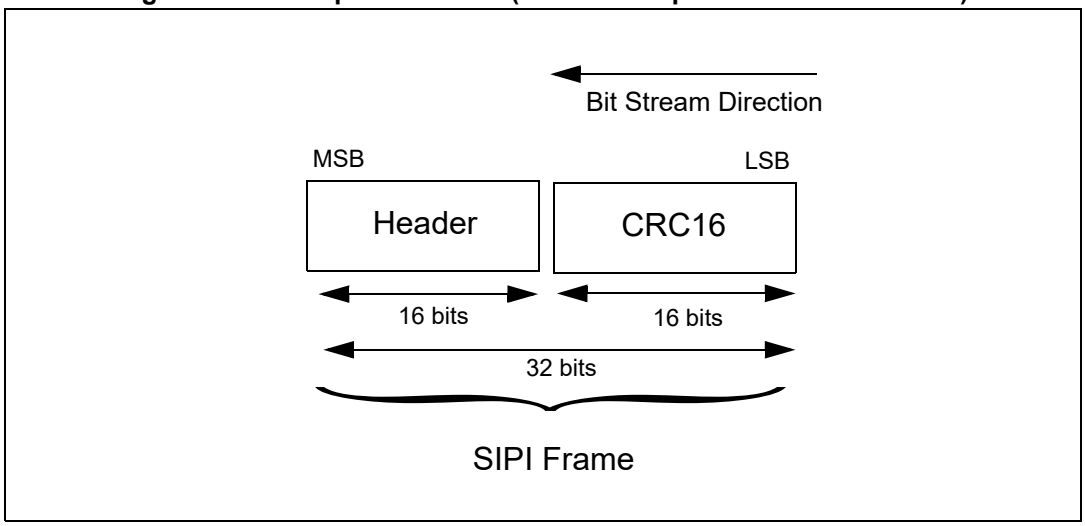


45.7.5 ID request response

45.7.5.1 ID request transfer

An ID request is transmitted by the initiator node as shown in *Figure 568*.

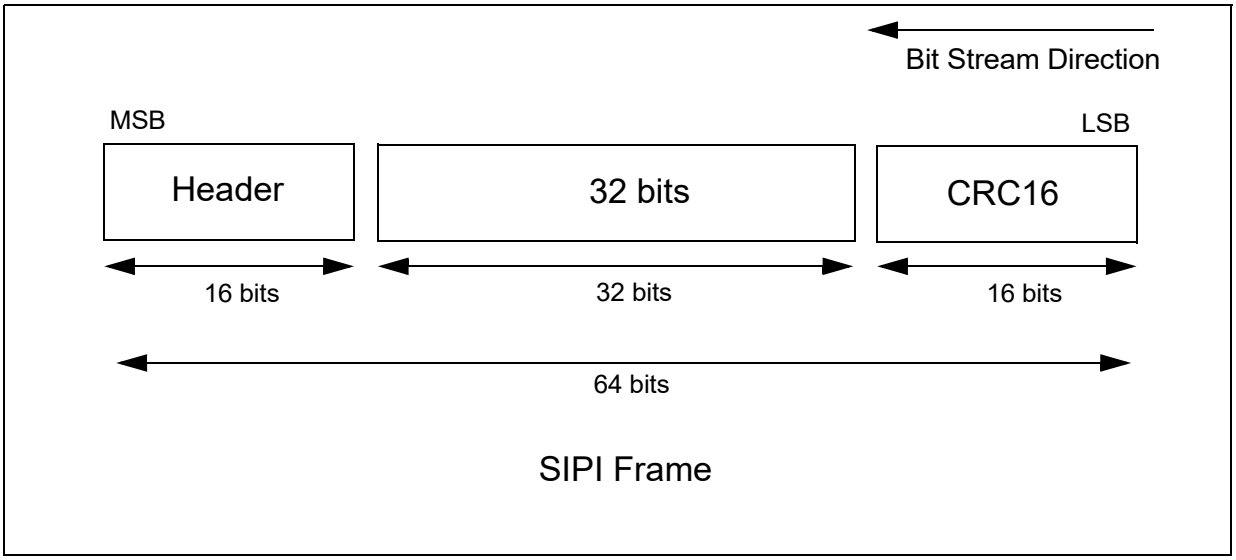
Figure 568. ID request transfer (LFAST encapsulation is not shown)



45.7.5.2 ID request response transfer

An ID request response is transmitted by the target node as shown in *Figure 569*.

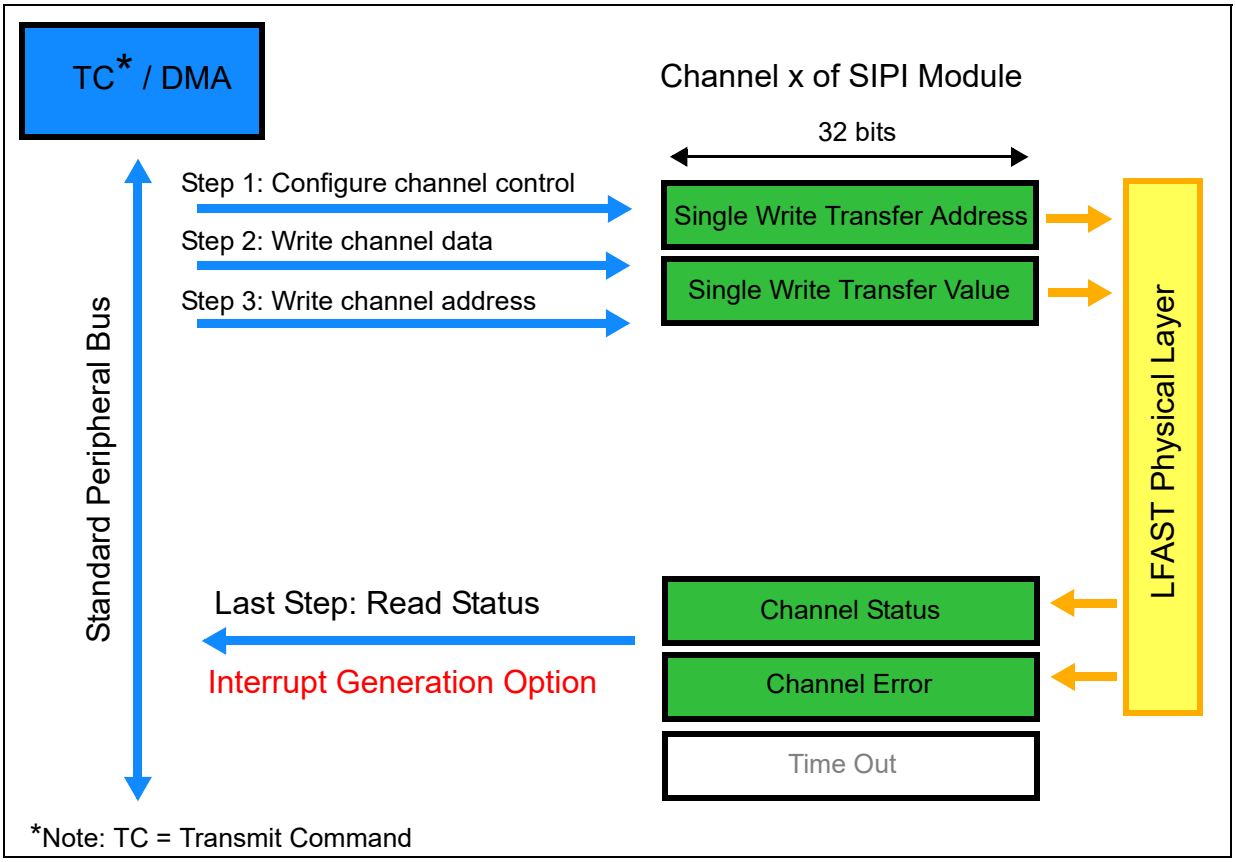
Figure 569. ID request response transfer



Note: The ID request response contains the value of the CHIP ID in place of data.

45.8 Transfer API and flow charts

Figure 570. SIPI single register write API



For a single Write Transfer Request generation (*Figure 570*):

1. Configure data and SIPI_CCR*n* at the initiator node.
2. Configure SIPI_CAR*n* at the initiator node.

   As soon as the channel address register (SIPI_CAR*n*) is written and if SIPI_CCR*n*[CHEN] = 1, the initiator SIPI will calculate the CRC on header, address and data field and will start transmitting data to LFAST.
3. Software polls the status register bits (CSR*n*) to determine when the request completes, SIPI_CSR*n*[ACKR] = 1. An interrupt will be generated if the corresponding SIPI_CIR*n*[ACKIE] = 1.

On a single Write transfer request reception (*Figure 571*):

1. Target node will place the address, data and control information on its AHB master interface.
2. When the process in completed, the target node will generate an acknowledge frame and send it back to the LFAST.

Figure 571. SIPI Single Register Write API – Flow Chart

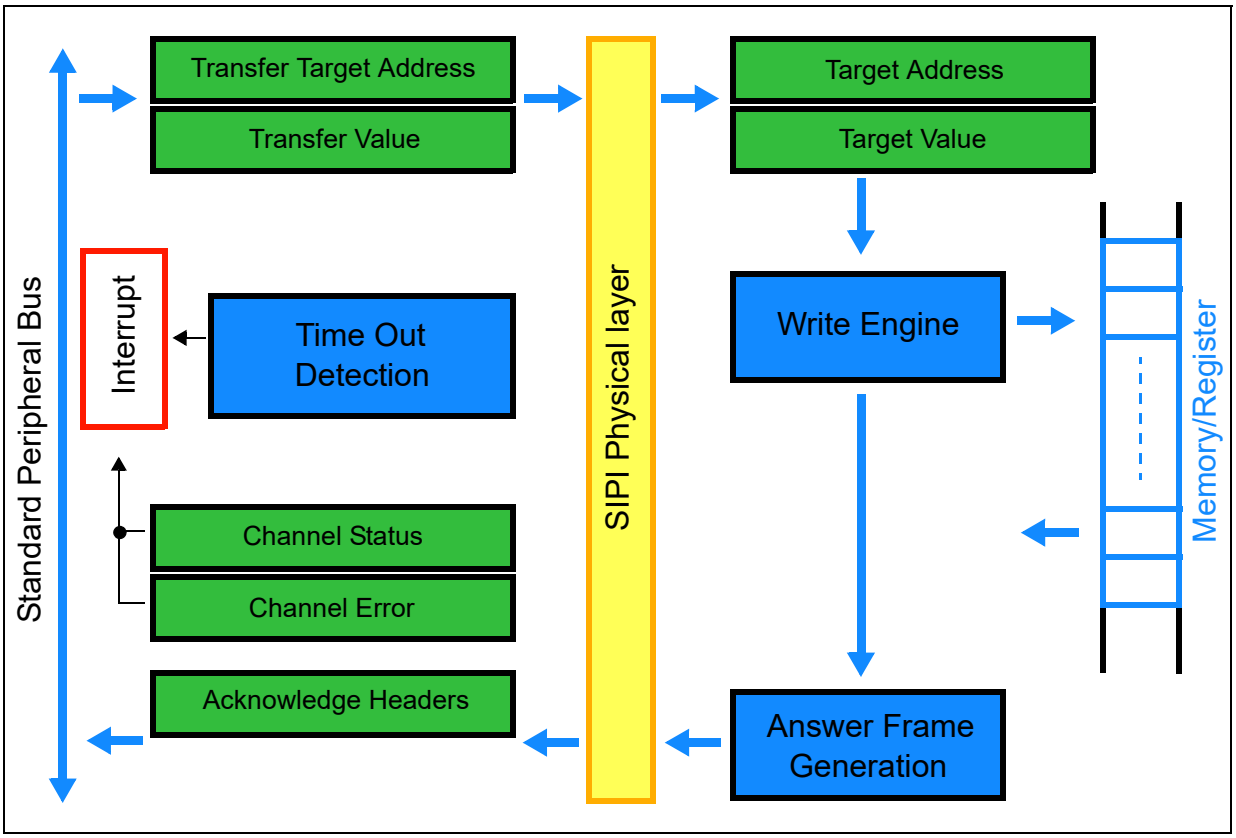
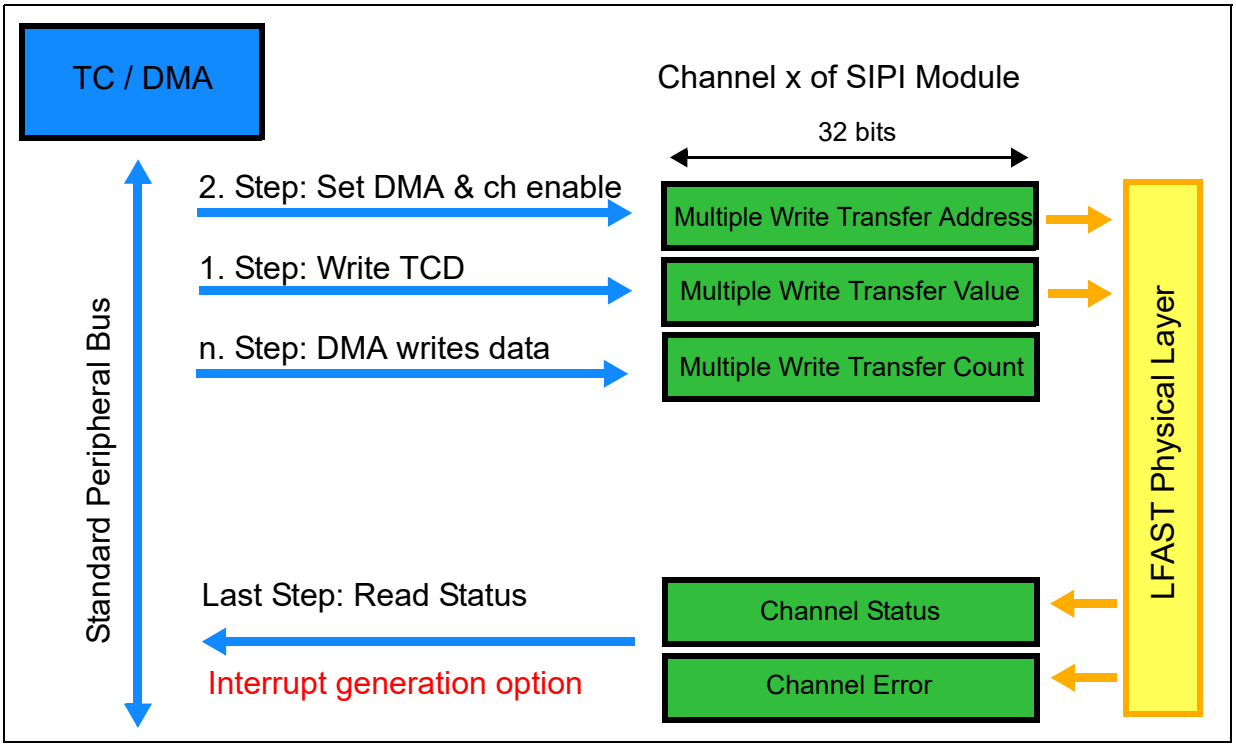


Figure 572. SIPI Multiple Register Write API



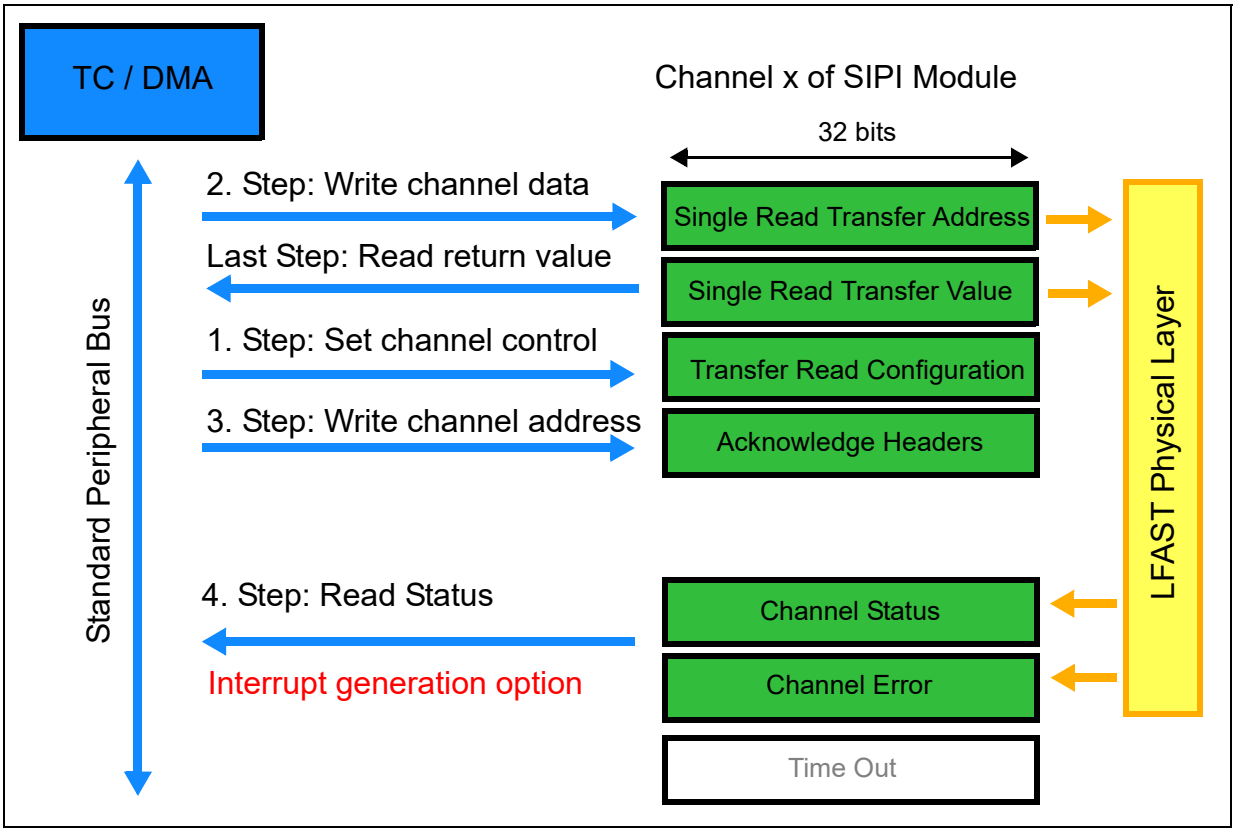
For Multiple Write Transfer Request generation (*Figure 572*):

1. Software will configure the Transfer Control Descriptor (TCD) of the DMA.
2. Software will write SIPI_CCR*n*[CHEN] = 1 and SIPI_CCR*n*[DAN] = 1.
3. SIPI will generate ipd_req and send it to the DMA controller.
4. SIPI will start copying data into the SIPI_CDR*n* through its DMA interface, depending on the transfer count and data registers size. SIPI_CDR2[0] should be written with the MSB.
5. When the copying process is completer, initiator SIPI will calculate CRC on header, address and data field and start transmitting data to LFAST.
6. Software should poll the CSR*n* status register bits to determine if the request has completed. If SIPI_CSR*n*[ACKR] = 1, an interrupt will be generated (if the corresponding SIPI_CIR*n*[ACKIE] = 1).
7. If SIPI_CCR*n*[DEN] = 1, the SIPI request will transfer to the will generate an ipd_req and send it to the DMA controller. If not, the ipd_req will not be sent and the state machine go idle.
8. Steps 4–7 will be repeated.

On Multiple Write transfer request reception (*Figure 572*):

1. Target node will place the address, data and control information on its AHB Master Interface.
2. When the process in completed, target node will generate an acknowledge frame and send it back to the LFAST.

Figure 573. SIPI Single Register Read API



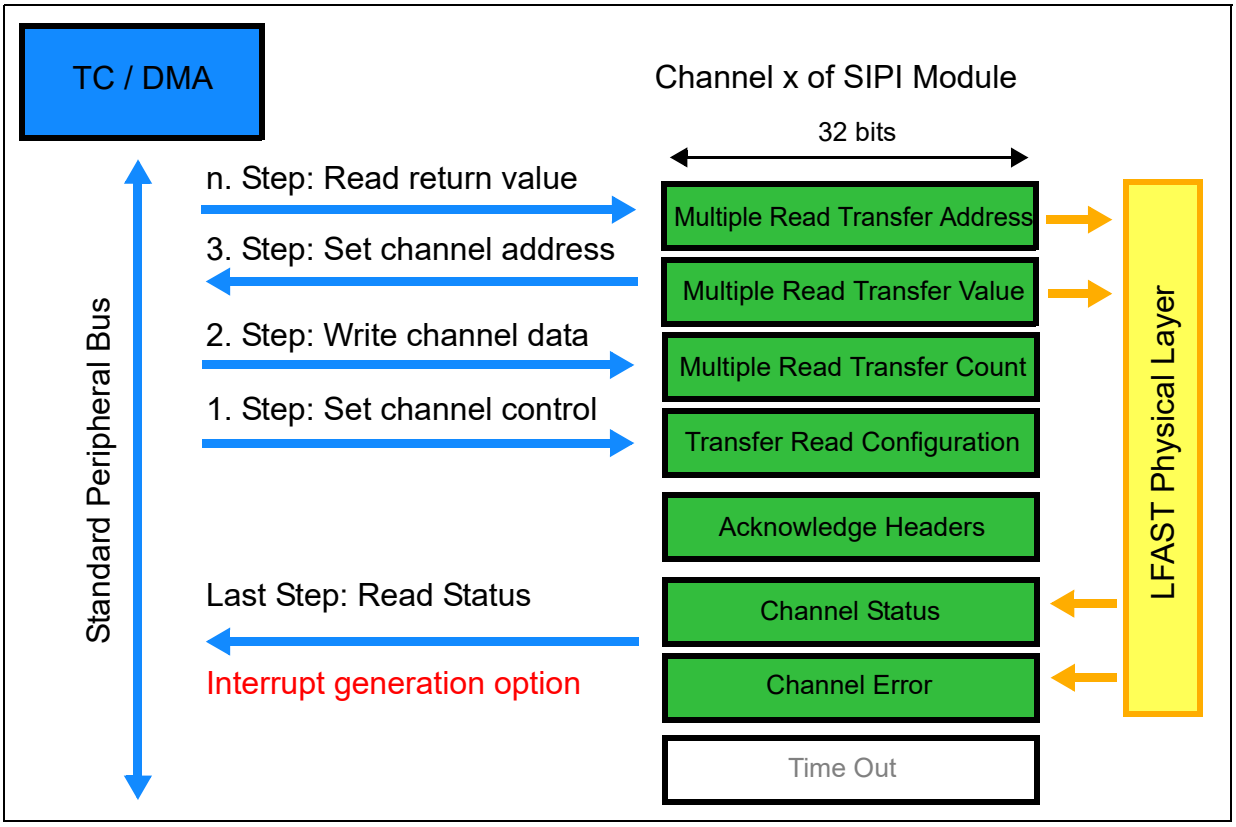
For Single Read Transfer Request generation (*Figure 573*):

1. Software/DMA will configure SIPI_CCR*n* and SIPI_CDR*n,* with the last step by writing SIPI_CAR*n*.
2. As soon as SIPI_CAR is written, the initiator SIPI will calculate CRC of the header and address fields and start transmitting data to the LFAST.
3. Software should poll SIPI_CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1, and interrupt will be generated (if SIPI_CIR*n*[RAIE] = 1). If SIPI_CSR*n*[RAR] does not set then SIPI_ERR[TOE*n*] = 1, which indicates a timeout has occurred.
4. If SIPI_CSR*n*[RAR] = 1, then software can read the data register, or can take other necessary action if SIPI_CSR*n*[RAR] = 0.

On Single Read transfer request reception (*Figure 573*):

1. Target node will place the address and control information on its AHB Master Interface.
2. When the process in completed, target node will send read response back to LFAST.

Figure 574. SIPI multiple register read API



For multiple read transfer request generation (*Figure 574*):

1. Software/DMA will configure SIPI_CCR*n* and SIPI_CDR*n,* with the last step by writing SIPI_CAR*n*.
2. As soon as channel address register is written, initiator SIPI will calculate CRC on the header and address fields, then start transmitting data to the LFAST.
3. Software should poll SIPI_CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1 and SIPI_CIR*n*[RAIE] = 1, an interrupt will be generated. If SIPI_CSR*n*[RAR] does not set, then SIPI_ERR[TOE*n*] = 1, which indicates a timeout.
4. Steps 2–3 will be repeated for multiple requests.

On Multiple Read transfer request reception (*Figure 574*):

1. Target node then will start reading data through its AHB interface.
2. When the transfer is completed/all data registers are full, it will calculate CRC and send the response frame back to LFAST.

Figure 575, *Figure 576*, *Figure 577* and *Figure 578* show the SIPI data streaming of data.

Figure 575. SIPI data stream model

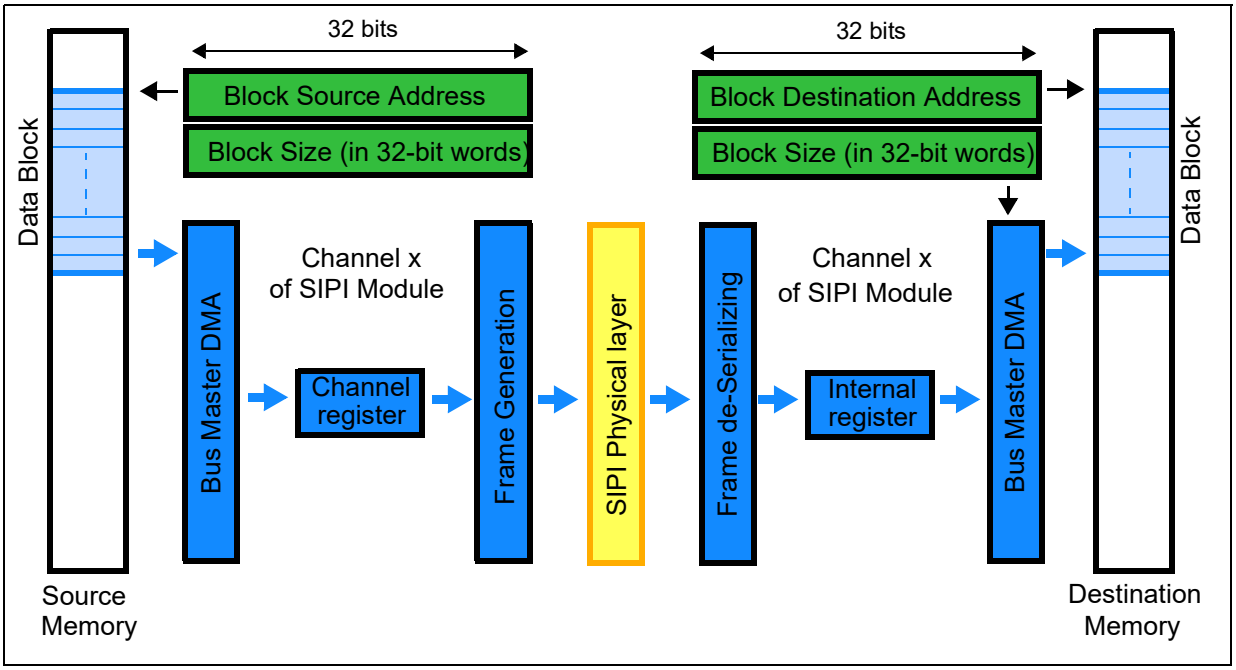


Figure 576. SIPI data stream model - Initiator

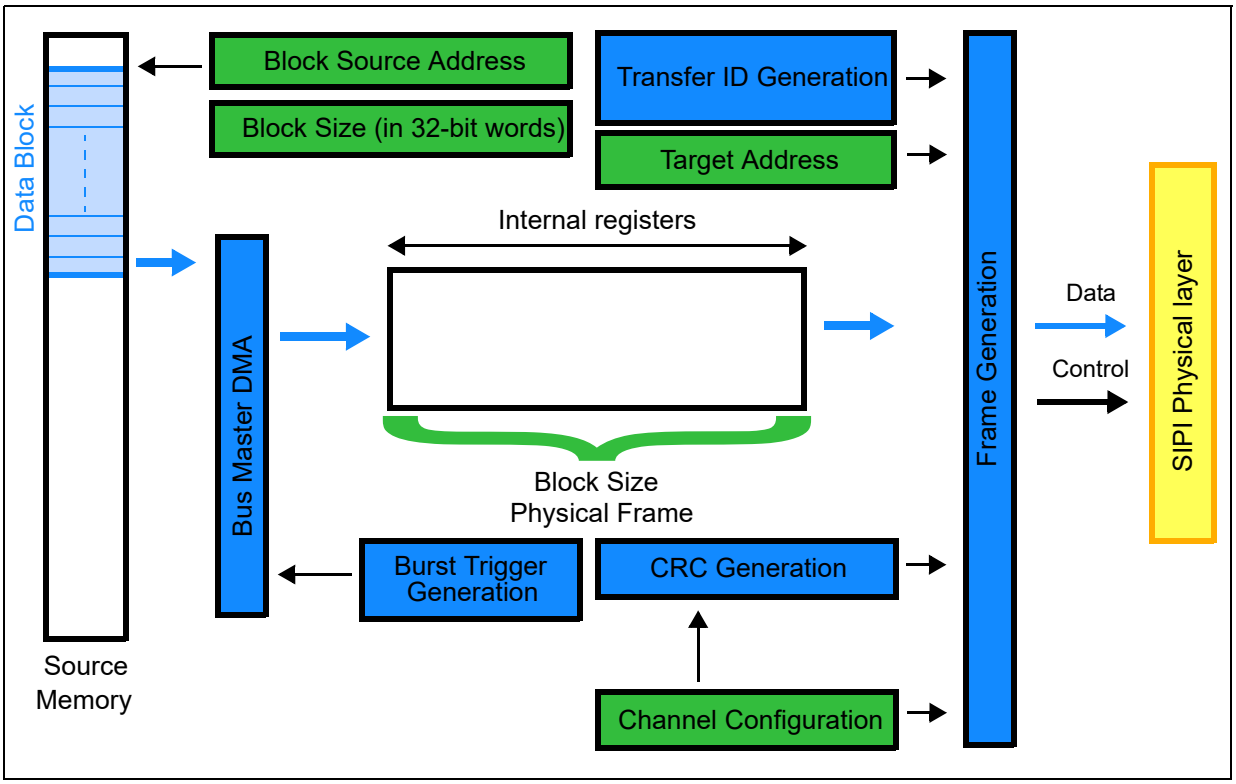


Figure 577. SIPI data stream model - Target

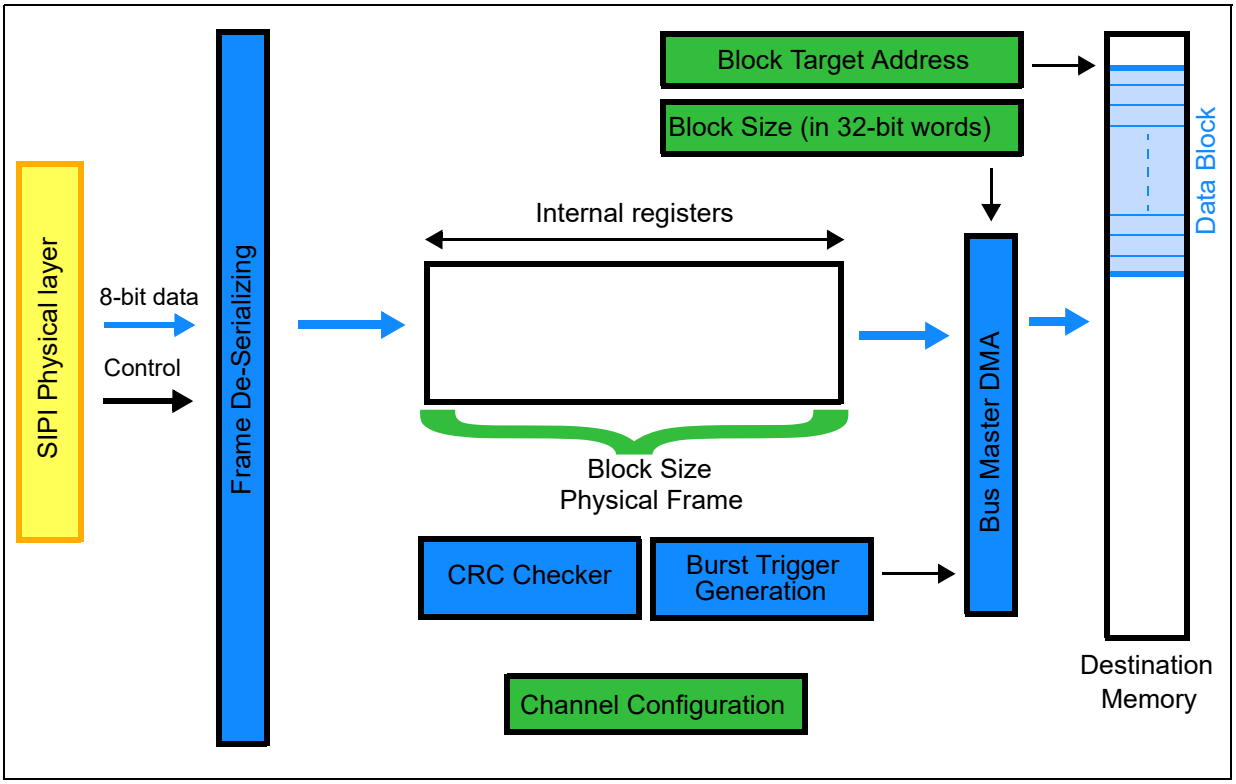
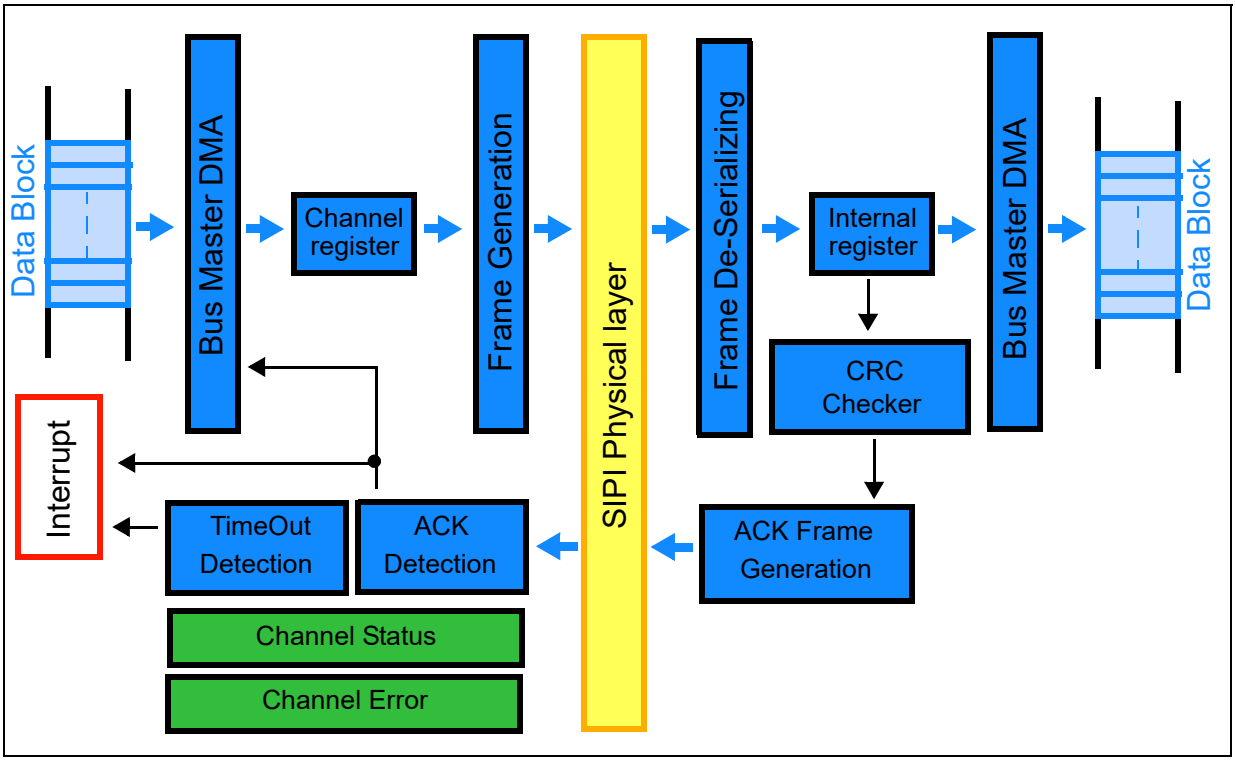


Figure 578. SIPI data stream with acknowledge trigger



45.9 DMA programming sequence

1. Software configures TCD (DMA data).
2. DMA transfers from RAM to SIPI registers.
3. SIPI starts transferring the data through LFAST Tx ports.
4. If an acknowledge or fault response is received, SIPI will repeat the process. If any error response is received, SIPI will move to an idle state. An interrupt will be flagged by SIPI, and software will take control of the SIPI.
5. If SIPI has received a read response, and error flags are not set on the LFAST Rx port, SIPI will assert a request. In case of an error, move to Step 8.
6. As soon as the request is asserted, DMA transfer starts between the SIPI registers and RAM.
7. If DMA transfer minor loop/major loop is complete, the request is negated.
8. If an error occurs:
   – The SIPI generates an interrupt
   – Software takes control of the SIPI

Steps 5–8 are repeated until the transfer is finished.

45.10 Modes of operation

SIPI has three operating modes:

- Initialization
- Normal
- Module Disable

After hardware reset the SIPI is in Module Disable mode, which helps reduce power consumption.

45.10.1 Initialization mode (INIT)

To enter Initialization mode software writes SIPI_MCR[INIT] = 1 (SIPI_MCR[MOEN] = 1 must be written before attempting to write SIPI_MCR[INIT]). To exit Initialization mode software writes SIPI_MCR[INIT] = 0 (see *Section 45.15.1.8: SIPI Module Configuration Register (SIPI_MCR)*).

Note: It is recommended that software checks the SIPI state (SIPI_MCR[MOEN]) before setting SIPI_MCR[INIT].

45.10.2 Normal mode (NM)

Once software has completed initialization of SIPI (SIPI_MCR[INIT] = 1), it can enter Normal mode by writing SIPI_MCR[INIT] = 0. SIPI needs to be in Normal mode for all data transfers (see *Section 45.15.1.8: SIPI Module Configuration Register (SIPI_MCR)*).

Note: SIPI must be enabled before attempting to write to SIPI_MCR[INIT] (for example, SIPI_MCR[MOEN] = 1).

45.10.3 Module disable (MD)

Module Disable mode in the SIPI is used to help reduce power consumption. By default, SIPI is in disable mode, SIPI_MCR[MOEN] = 0, and is exited by writing SIPI_MCR[MOEN] = 1 (see *Section 45.15.1.8: SIPI Module Configuration Register (SIPI_MCR)*). Future SIPI Tx transfers are disabled in Module Disable mode. To disable Rx functionality, the Target Enable (TEN) bit field has to be deasserted, then the internal Rx state machine resets to initial state.

45.11 Errors

This section describes the potential errors that can occur during SIPI operation.

45.11.1 Timeout error

This error is generated at the initiator node when the acknowledge/response is not received within the configured time in the corresponding SIPI_CTOR*n*[TOR] field setting (see *Section 45.15.1.4: SIPI Channel Timeout Register (SIPI_CTORn)*). SIPI_ERR[TOE*n*] = 1 indicates that a channel timeout error has occurred (see *Section 45.15.1.13: SIPI Channel Error Register (SIPI_ERR)*).

Note: SIPI should not drop the response even after timeout occurs. Software polls both error and status flags after the transfer to see if there was a timeout error. If there was a timeout error the response received may be discarded.

45.11.2 CRC error

A CRC error is generated at the target nodes when the CRC received with the frame does not match with the calculated CRC. When a CRC error is detected the SIPI_SR[GCRCE] is set (see *Section 45.15.1.9: SIPI Status Register (SIPI_SR)*) and an interrupt will be generated if SIPI_MCR[CRCIE] = 1 (*Section 45.15.1.8: SIPI Module Configuration Register (SIPI_MCR)*).

Note: In the case of a CRC error, the target node will not send an acknowledge to the initiator node. Then an interrupt will be generated on the target side if the corresponding interrupt enable bit is set. The initiator node will detect a time-out and take necessary action.

45.11.3 Maximum count reached error

This error is always generated at the target node only. The maximum count reached error is generated when the value of the SIPI_ACR equals SIPI_MAXCR (see *Section 45.15.1.12: SIPI Address Count Register (SIPI_ACR)* and *Section 45.15.1.10: SIPI Max Count Register (SIPI_MAXCR)* for more details). The SIPI_SR[MCR] will be set (see *Section 45.15.1.9: SIPI Status Register (SIPI_SR)*), and an interrupt will be generated if SIPI_MCR[MCRIE] = 1 (*Section 45.15.1.8: SIPI Module Configuration Register (SIPI_MCR)*).

45.11.4 Transaction ID error

This error is always generated at the initiator node only. It is generated when header bits 15–13 do not match SIPI_CSR*n*[TID] (transaction ID bits, see *Section 45.15.1.2: SIPI Channel Status Register (SIPI_CSRn)*). The SIPI_CSR*n*[TIDE] bit will be set when a TID error is detected, and an interrupt will be generated if SIPI_CIR*n*[TIDIE] = 1 (see *Section 45.15.1.3: SIPI Channel Interrupt Register (SIPI_CIRn)*).

45.11.5 Acknowledge error

When and incorrect Acknowledge is received the SIPI_ERR[ACKR*n*] will be set (see *Section 45.15.1.13: SIPI Channel Error Register (SIPI_ERR)*), at the initiator node only, and an interrupt will be generated if SIPI_CIR*n*[WAIE] = 1 (see *Section 45.15.1.3: SIPI Channel Interrupt Register (SIPI_CIRn)*).

45.12 CRC calculation

Example: If header is AABBh, address is 1122_3344h and data is CCDD_EEFFh. Then CRC calculation will take place as follows:

1) The CRC seed is initialized by FFFF_FFFFh.

2) All the data will be mirrored before sending to CRC engine (for example, MSB will be sent as LSB).

3) So the header will be sent as DD55_0000h, byte_enable = Ch.

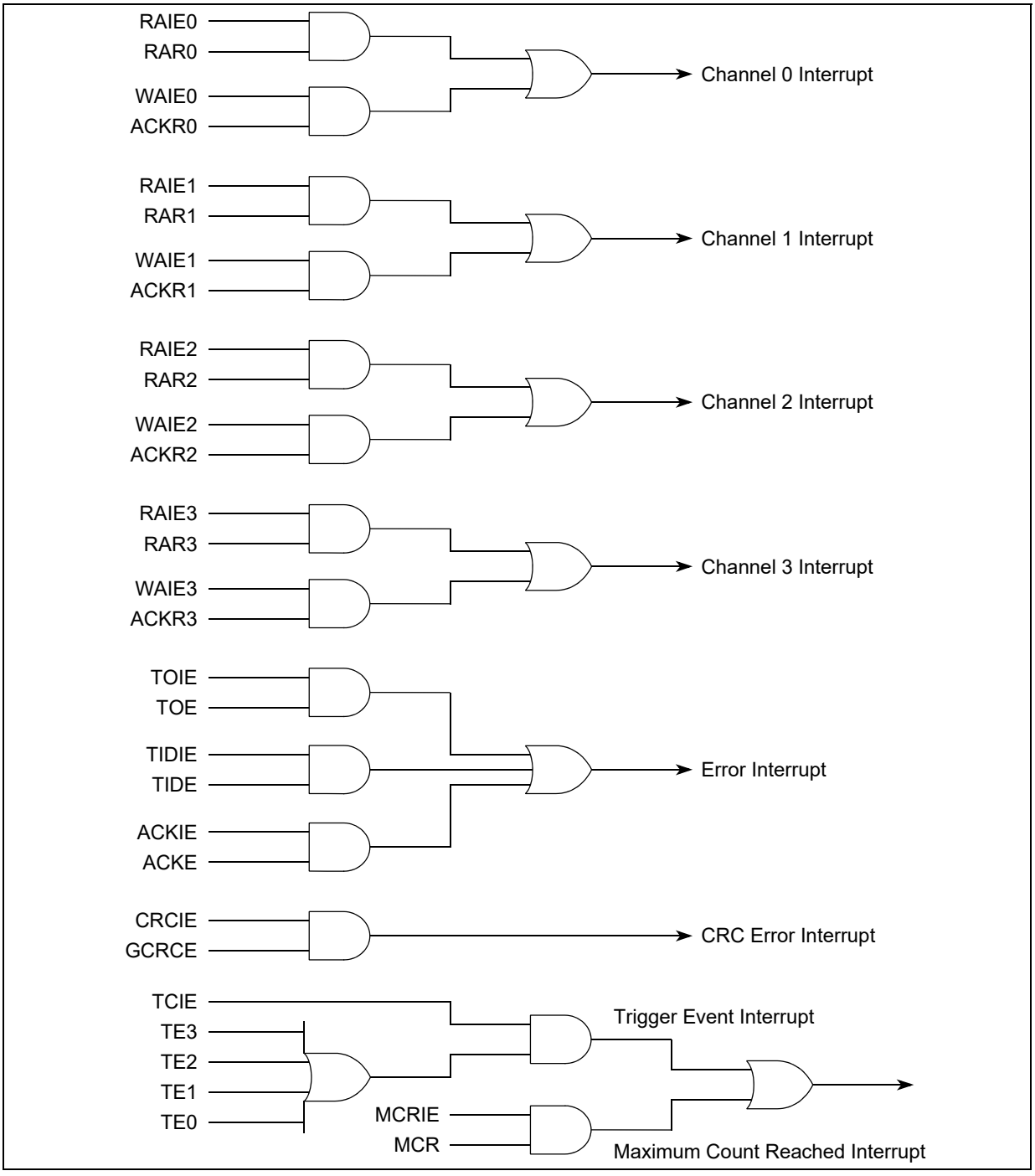
4) Address will be sent as 22CC_4488h, byte_enable = Fh.

5) Data will be sent as FF77_BB33h, byte enable = Fh.

45.13 Interrupt logic

A description of the interrupt logic can be found in *Figure 579*.

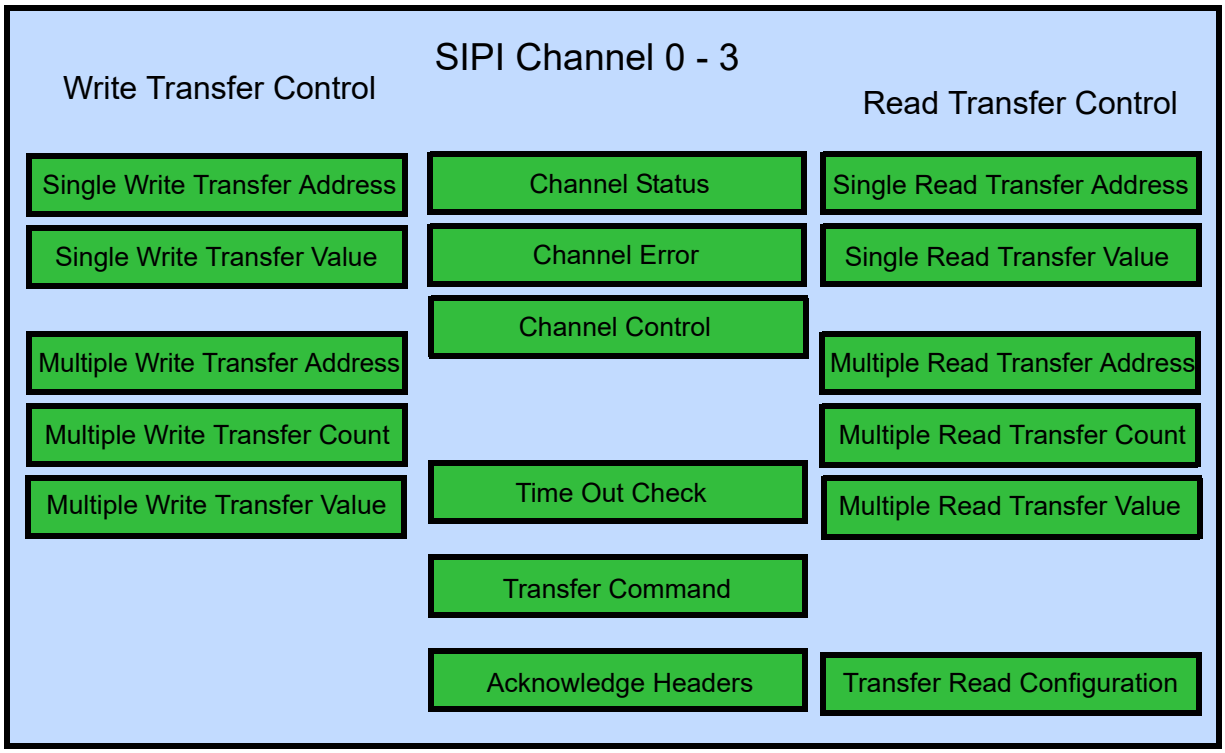
Figure 579. Interrupt description



45.14 SIPI control and status overview

The diagram below shows the relationship between transfers and the SIPI control and status registers.

Figure 580. SIPI control & status overview – Register transfer



45.15 Memory map and register description

Table 613. SIPI memory map

Offset from SIPI base (hex)	Register	Section/Page
0000	SIPI Channel Control Register 0 (SIPI_CCR0)	*Section 45.15.1.1*
0004	SIPI Channel Status Register 0 (SIPI_CSR0)	*Section 45.15.1.2*
0008	Reserved	—
000C	SIPI Channel Interrupt Register 0 (SIPI_CIR0)	*Section 45.15.1.3*
0010	SIPI Channel Timeout Register 0 (SIPI_CTOR0)	*Section 45.15.1.4*
0014	SIPI Channel CRC Register 0 (SIPI_CCRC0)	*Section 45.15.1.5*
0018	SIPI Channel Address Register 0 (SIPI_CAR0)	*Section 45.15.1.6*
001C	SIPI Channel Data Register 0 (SIPI_CDR0)	*Section 45.15.1.7*
0020	SIPI Channel Control Register 1 (SIPI_CCR1)	*Section 45.15.1.1*
0024	SIPI Channel Status Register 1 (SIPI_CSR1)	*Section 45.15.1.2*
0028	Reserved	—
002C	SIPI Channel Interrupt Register 1 (SIPI_CIR1)	*Section 45.15.1.3*
0030	SIPI Channel Timeout Register 1 (SIPI_CTOR1)	*Section 45.15.1.4*

Table 613. SIPI memory map(Continued)

Offset from SIPI base (hex)	Register	Section/Page
0034	SIPI Channel CRC Register 1 (SIPI_CCRC1)	*Section 45.15.1.5*
0038	SIPI Channel Address Register 1 (SIPI_CAR1)	*Section 45.15.1.6*
003C	SIPI Channel Data Register 1 (SIPI_CDR1)	*Section 45.15.1.7*
0040	SIPI Channel Control Register 2 (SIPI_CCR2)	*Section 45.15.1.1*
0044	SIPI Channel Status Register 2 (SIPI_CSR2)	*Section 45.15.1.2*
0048	Reserved	—
004C	SIPI Channel Interrupt Register 2 (SIPI_CIR2)	*Section 45.15.1.3*
0050	SIPI Channel Timeout Register 2 (SIPI_CTOR2)	*Section 45.15.1.4*
0054	SIPI Channel CRC Register 2 (SIPI_CCRC2)	*Section 45.15.1.5*
0058	SIPI Channel Address Register 2 (SIPI_CAR2)	*Section 45.15.1.6*
005C	SIPI Channel Data Register 2[0] (SIPI_CDR2[0])	*Section 45.15.1.7*
0060	SIPI Channel Data Register 2[1] (SIPI_CDR2[1])	*Section 45.15.1.7*
0064	SIPI Channel Data Register 2[2] (SIPI_CDR2[2])	*Section 45.15.1.7*
0068	SIPI Channel Data Register 2[3] (SIPI_CDR2[3])	*Section 45.15.1.7*
006C	SIPI Channel Data Register 2[4] (SIPI_CDR2[4])	*Section 45.15.1.7*
0070	SIPI Channel Data Register 2[5] (SIPI_CDR2[5])	*Section 45.15.1.7*
0074	SIPI Channel Data Register 2[6] (SIPI_CDR2[6])	*Section 45.15.1.7*
0078	SIPI Channel Data Register 2[7] (SIPI_CDR2[7])	*Section 45.15.1.7*
007C	SIPI Channel Control Register 3 (SIPI_CCR3)	*Section 45.15.1.1*
0080	SIPI Channel Status Register 3 (SIPI_CSR3)	*Section 45.15.1.2*
0084	Reserved	—
0088	SIPI Channel Interrupt Register 3 (SIPI_CIR3)	*Section 45.15.1.3*
008C	SIPI Channel Timeout Register 3 (SIPI_CTOR3)	*Section 45.15.1.4*
0090	SIPI Channel CRC Register 3 (SIPI_CCRC3)	*Section 45.15.1.5*
0094	SIPI Channel Address Register 3 (SIPI_CAR3)	*Section 45.15.1.6*
0098	SIPI Channel Data Register 3 (SIPI_CDR3)	*Section 45.15.1.7*
009C	SIPI Module Configuration Register (SIPI_MCR)	*Section 45.15.1.8*
00A0	SIPI Status Register (SIPI_SR)	*Section 45.15.1.9*
00A4	SIPI Max Count Register (SIPI_MAXCR)	*Section 45.15.1.10*
00A8	SIPI Address Reload Register (SIPI_ARR)	*Section 45.15.1.11*
00AC	SIPI Address Count Register (SIPI_ACR)	*Section 45.15.1.12*
00B0	SIPI Error Register (SIPI_ERR)	*Section 45.15.1.13*

45.15.1 Register descriptions

45.15.1.1 SIPI Channel Control Register (SIPI_CCR*n*)

Note: Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSRn[CB] = 1, is configured for 1 channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSRn[CB] = 0).

Note: PRIORITY SCHEDULING: The channel whose SIPI_CARn is written first will have access first, irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Priority scheduling occurs when the LFAST is unable to receive data, and no channel is pending transmission. The reason for the scheduling is that a common CRC module is shared by all channels.

Note: This register can only be written during Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSRn[CB] = 1, Section 45.15.1.2: SIPI Channel Status Register (SIPI_CSRn)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Note: Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CARn (Section 45.15.1.6: SIPI Channel Address Register (SIPI_CARn).

Offset: 0000h (CCR0), 0020h (CCR1), 0040h (CCR2), 007Ch (CCR3)

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	TC(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	WL[1:0]		CHEN	ST(2)	IDT(1)	RRT(1)	WRT(1),(2)	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The order of priority for writing to IDT, WRT and RRT must be in this order: 1) IDT, 2) RRT, 3) WRT.
2. Only transfers with channel 2 can write 1 to this bit. All other channels forces SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming.

Figure 581. SIPI Channel Control Register (SIPI_CCR*n*)

Table 614. SIPI_CCR*n* field descriptions

Field	Description
0–14	Reserved

Table 614. SIPI_CCR*n* field descriptions(Continued)

Field	Description
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in the SIPI_CAR. 0 Trigger command not sent 1 Trigger command sent
16–23	Reserved
24–25 WL[1:0]	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all the channel enables are enabled simultaneously, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0 for channels 0, 1 and 3. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when SIPI_CCR*n*[WRT] = 1. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. 0 Read request will not be sent 1 Read request transfer by the initiator. This bit cannot be written if SIPI_CCR*n*[IDT] = 1.
30 WRT	Write Request Transfer. 0 No write request will be sent 1 Write request transfer by the initiator. This bit cannot be written if SIPI_CCR*n*[IDT] = 1 or SIPI_CCR*n*[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access 1 Channel will be used for DMA access

45.15.1.2 SIPI Channel Status Register (SIPI_CSR*n*)

The SIPI_CSR*n* contains the status bits for the current transfer.

Offset: 0004h (CSR0)
0024h (CSR1)
0044h (CSR2)
0080h (CSR3)

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	RAR	TID[2:0]			ACKR	CB	0	0
W									w1c	w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 582. Channel Status Register (CSR*n*)

Table 615. CSR*n* field descriptions

Field	Description
0–23	Reserved
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID[2:0]	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel status. This bit is read only. 0 Channel free 1 Channel busy
30–31	Reserved

45.15.1.3 SIPI Channel Interrupt Register (SIPI_CIR*n*)

The SIPI_CIR*n* contains the interrupt enable bits for the corresponding channel (*n*).

Offset: 000Ch (CIR0), 002Ch (CIR1), 004Ch (CIR2), 0088h (CIR3)

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	WAIE	RAIE	TCIE	TOIE	TIDIE	ACKIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 583. SIPI Channel Interrupt Register (SIPI_CIR*n*)

Table 616. SIPI_CIR*n* field descriptions

Field	Description
0–25	Reserved
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled. 0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

45.15.1.4 SIPI Channel Timeout Register (SIPI_CTOR*n*)

The SIPI_CTOR*n* contains the timeout value for Tx requests.

Offset: 0010h (CTOR0), 0030h (CTOR1), 0050h (CTOR2), 008Ch (CTOR3)

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	TOR(1)							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

1. Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

Figure 584. SIPI Channel Timeout Register (SIPI_CTOR*n*)

Table 617. SIPI_CTOR*n* field descriptions

Field	Description
0–23	Reserved
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. These bits can only be written by software in initialization mode, SIPI_MCR[INIT] = 1.

45.15.1.5 SIPI Channel CRC Register (SIPI_CCRC*n*)

SIPI_CCRC*n* is read-only register that returns the CRC calculated value on the header, address, and data bits.

Offset: 0014h (CCRC0), 0034h (CCRC1), 0054h (CCRC2), 0090h (CCRC3)

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRCI[31:16]															
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	CRCT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 585. SIPI Channel CRC Register (SIPI_CCRC*n*)

Table 618. SIPI_CCRC*n* field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

45.15.1.6 SIPI Channel Address Register (SIPI_CAR*n*)

The SIPI_CAR*n* contains the data transmission target address. For streaming operations this register contains the start address. The SIPI_CAR*n* is a 32-bit register and can be written anytime by software or DMA.

Offset: 0018h (CAR0)
0038h (CAR1)
0058h (CAR2)
0094h (CAR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CAR[31:16](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	CAR[15:0](1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This register holds the start address of streaming operations.

Figure 586. SIPI Channel address Register (SIPI_CAR*n*)

Table 619. SIPI_CAR*n* field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

45.15.1.7 SIPI Channel Data Register (SIPI_CDR*n*)

The SIPI_CDR*n* contains the data to be transmitted, and can be written anytime by software or DMA. The data can be written in 8, 16 or 32-bit formats.

Offset: See *Table 621* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CDR[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	CDR[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 587. SIPI Channel Data Register (SIPI_CDR*n*)

Table 620. SIPI_CDR*n* field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

Table 621. Channel data Register address offsets

Register	Address Offset (hex)
CDR0	001Ch
CDR1	003Ch
CDR2[0]–CDR2[7]	005Ch, 0060h, 0064h, 0068h, 006Ch, 0070h, 0074h, 0078h
CDR3	0098h

45.15.1.8 SIPI Module Configuration Register (SIPI_MCR)

The SIPI_MCR is a 32-bit global configuration register of the SIPI.

Offset: 009Ch Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZ	Reserved	HALT	0	0	PRSCLR(1)										
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AID(1)		0	0	0	CRCIE(2)	MCRIE(2)	0	0	0	0	CHNSB(3)	TEN(2)	INIT(2)	MOEN	SR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Can be written in initialization mode only (SIPI_MCR[INIT] = 1).
2. Can only be written when SIPI_MCR[MOEN] = 1.
3. Can be written only once after reset.

Figure 588. SIPI Module Configuration Register (SIPI_MCR)

Table 622. SIPI_MCR field descriptions

Field	Description
0 FRZ	Freeze Enable The FRZ bit specifies the SIPI behavior when Debug mode is requested at the MCU level. When FRZ is asserted, the SIPI is enabled to enter Freeze mode. Negation of this bit field causes SIPI to exit Freeze mode. 0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
1	Reserved
2 HALT	Halt Mode Enable. Assertion of this bit puts SIPI into Freeze mode. No Rx or Tx is performed in the SIPI until this bit is cleared. If this bit is enabled in during Tx or Rx communications, the current activity will finish, then the SIPI will enter Freeze mode. 0 No Freeze mode request 1 Enters Freeze mode if FRZ bit is asserted.
3–4	Reserved
5–15 PRSCLR	These bits should be programmed by software in initialization mode, SIPI_MCR[INIT] = 1, writes during other times are ignored. The timeout counter runs on the prescaled system clock. Default value is 64 (040h). The allowed programmable values are: 040h 64 080h 128 100h 256 200h 512 400h 1024
16–17 AID	Address Increment/Decrement. These bits define the type of address change at the target node. 00 no change. address stays same 01 address increments by 4 10 address decrements by 4. 11 Reserved
18–20	Reserved

Table 622. SIPI_MCR field descriptions(Continued)

Field	Description
21 CRCIE	CRC Error Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
22 MCRIE	Max Count Reached Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
23–26	Reserved
27 CHNSB	Channel coding select bit 0, 1 Channel number select as per *Figure 610: SIPI Header Channel Number Coding*.
28 TEN	Target Enable. Setting this bit enables the target functionality at SIPI. This bit can be read or written anytime. This bit is automatically negated by hardware when target detects an error in “streaming without ACK” mode. This bit has to be enabled for transmission operations as well.
29 INIT	Setting this bit puts the module in initialization mode. This bit should be cleared by software. Most register bits are configured using this bit. The SIPI_MCR[MOEN] bit needs to be set first, and then the INIT bit can be set and both bits cannot be enabled together. 0 Normal Mode 1 Initialization Mode
30 MOEN	Module Enable. This bit should be set or cleared by software. When this bit is negated, future SIPI Tx transfers are disabled.
31 SR	Soft Reset. Setting this bit clears all status and error registers and FSMs are moved to idle state. This bit is automatically cleared by hardware once the reset operation is complete.

45.15.1.9 SIPI Status Register (SIPI_SR)

The SIPI_SR is the global status register of SIPI.

Offset: 00A0h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZACK	LPMACK	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	GCRCE	MCR	0	TE[3:0]				STATE			
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 589. SIPI Status Register (SIPI_SR)

Table 623. SIPI_SR field descriptions

Field	Description
0 FRZACK	Freeze Mode Acknowledge. This read-only bit indicates that SIPI is in Freeze mode. The Freeze mode request cannot be granted until current transmission or reception processes have finished. The software can poll the FRZACK bit to find when the SIPI has actually entered Freeze mode. If Freeze mode is requested while SIPI is in any of the low power modes, then the FRZACK bit will only be set when the low power mode is exited. 0 SIPI not in Freeze mode 1 SIPI in Freeze mode
1 LPMACK	Low Power Mode Acknowledge This read-only bit indicates that SIPI is in Disable Mode. Disable mode cannot be entered until all current transmission or reception processes have finished. The CPU can poll the LPMACK bit to know when SIPI has actually entered low power mode. 0 SIPI not in any low power modes 1 SIPI is in Disable mode
2–20	Reserved
21 GCRCE	Global CRC Error Bit 0 No CRC error 1 CRC error occurred
22 MCR	Maximum Count Reached. This bit will be set whenever SIPI_ACR[ADCNT] = SIPI_MAXCR[MXCNT]. An interrupt will be generated when this bit is set only if SIPI_MCR[MCRIE] = 1. This bit should be cleared by software.
23	Reserved
24–27 TE	Trigger Event on Respective Channels. This field enables interrupts for target nodes. TE3 = 1 - Channel 3 trigger event TE2 = 1 - Channel 2 trigger event TE1 = 1 - Channel 1 trigger event TE0 = 1 - Channel 0 trigger event
28–31 STATE	These bits reflect the transmit state machine status. They can be polled for determination of state machine status. 0000 IDLE 0001 HEADER_AND_ADDRESS_FIELD 0010 HEADER_AND_DATA_FIELD 0011 HEADER_AND_CRC_FIELD 0100 ADDRESS_AND_CRC_FIELD 0101 ADDRESS_AND_DATA_FIELD 0110 DATA_AND_CRC_FIELD 0111 DATA_FIELD

45.15.1.10 SIPI Max Count Register (SIPI_MAXCR)

The SIPI_MAXCR contains the maximum address count value at target node. It is programmed via direct write request through the initiator. This register can be read or written by software anytime.

Offset: 00A4h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MXCNT[31:16]															
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	MXCNT[15:2]														0	0
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

Figure 590. SIPI Max Count Register (SIPI_MAXCR)

Table 624. SIPI_MAXCR field descriptions

Field	Description
0–29 MXCNT	This field contains the maximum address count value at the target node. It should be programmed via direct write request through the initiator.
30-31	Reserved

45.15.1.11 SIPI Address Reload Register (SIPI_ARR)

The SIPI_ARR contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator. This register is writeable only when SIPI_MCR[INIT] = 1.

Offset: 00A8h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADRLD[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	ADRLD[15:2]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 591. SIPI Address Reload Register (SIPI_ARR)

Table 625. SIPI_ARR field descriptions

Field	Description
0–29 ADRLD	Contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.
30-31	Reserved

45.15.1.12 SIPI Address Count Register (SIPI_ACR)

This register reflects the count value of address counter at target node. It should be configured by direct write request from initiator. This register can be read/written by software anytime.

Offset: 00ACh Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADCNT[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	ADCNT[15:2]														0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 592. SIPI Address Count Register (SIPI_ACR)

Table 626. SIPI_ACR field descriptions

Field	Description
0–29 ADCNT	Reflects the count value of address counter at target node. It should be configured by direct write request from the initiator. This register can be read/written by software anytime. At the end of the streaming operation, SIPI_ACR will point to the next address to be written.
30-31	Reserved

Note: SIPI_ARR, SIPI_ACR and SIPI_MAXCR will be configured by direct write operation through the initiator.

Note: When a streaming write command is received, the target will start the write operation from the address present in SIPI_ACR.

Note: After each 32-bit write has completed, SIPI_ACR[ADCNT] is compared against SIPI_MAXCR[MXCNT]. If they are equal, SIPI_ACR[ADCNT] is loaded with the value stored in SIPI_ARR[ADRLD]. If they are not equal, the SIPI_ACR[ADCNT] will increment by 4, decrement by 4, or remain the same (depending on configuration). In either case, the SIPI_ACR[ADCNT] will contain the address at which the next streaming write will occur.

45.15.1.13 SIPI Channel Error Register (SIPI_ERR)

This register contains the error bits for the last transaction(s) for all the channels. Communication errors are generated only for out of range address accesses, areas that are read only and where accesses do not lead to generation of transfer errors. Also, writing to read only registers will not generate communication errors.

Offset: 00B0h 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	TOE3	TIDE3	ACKE3	0	0	0	0	0	TOE2	TIDE2	ACKE2
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	TOE1	TIDE1	ACKE1	0	0	0	0	0	TOE0	TIDE0	ACKE0
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 593. SIPI Channel Error Register (SIPI_ERR)

Table 627. SIPI_ERR field descriptions

Field	Description
0–4	Reserved
5 TOE3	Timeout Error for Channel 3. 0 Timeout error has occurred 1 Timeout error has not occurred
6 TIDE3	Transaction ID Error for Channel 3. 0 Received transaction ID matched with the stored ID 1 Received transaction ID did not match with the stored ID
7 ACKE3	Acknowledge Error for Channel 3. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
8-12	Reserved
13 TOE2	Timeout Error for Channel 2. 0 Timeout error has occurred 1 Timeout error has not occurred
14 TIDE2	Transaction ID Error for Channel 2. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn’t match with the stored ID
15 ACKE2	Acknowledge Error for Channel 2. 0 Acknowledge received is correct 1 Acknowledge received is not correct
16-20	Reserved
21 TOE1	Timeout Error for Channel 1. 0 Timeout error has occurred 1 Timeout error has not occurred
22 TIDE1	Transaction ID Error for Channel 1. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn’t match with the stored ID

Table 627. SIPI_ERR field descriptions(Continued)

Field	Description
23 ACKE1	Acknowledge Error for Channel 1. 0 Acknowledge received is correct 1 Acknowledge received is not correct
24–28	Reserved
29 TOE0	Timeout Error for Channel 0. 0 Timeout error has not occurred 1 Timeout error has occurred
30 TIDE0	Transaction ID Error for Channel 0. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn’t match with the stored ID
31 ACKE0	Acknowledge Error for Channel 0. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.

46 Deserial Serial Peripheral Interface (DSPI)

46.1 Introduction

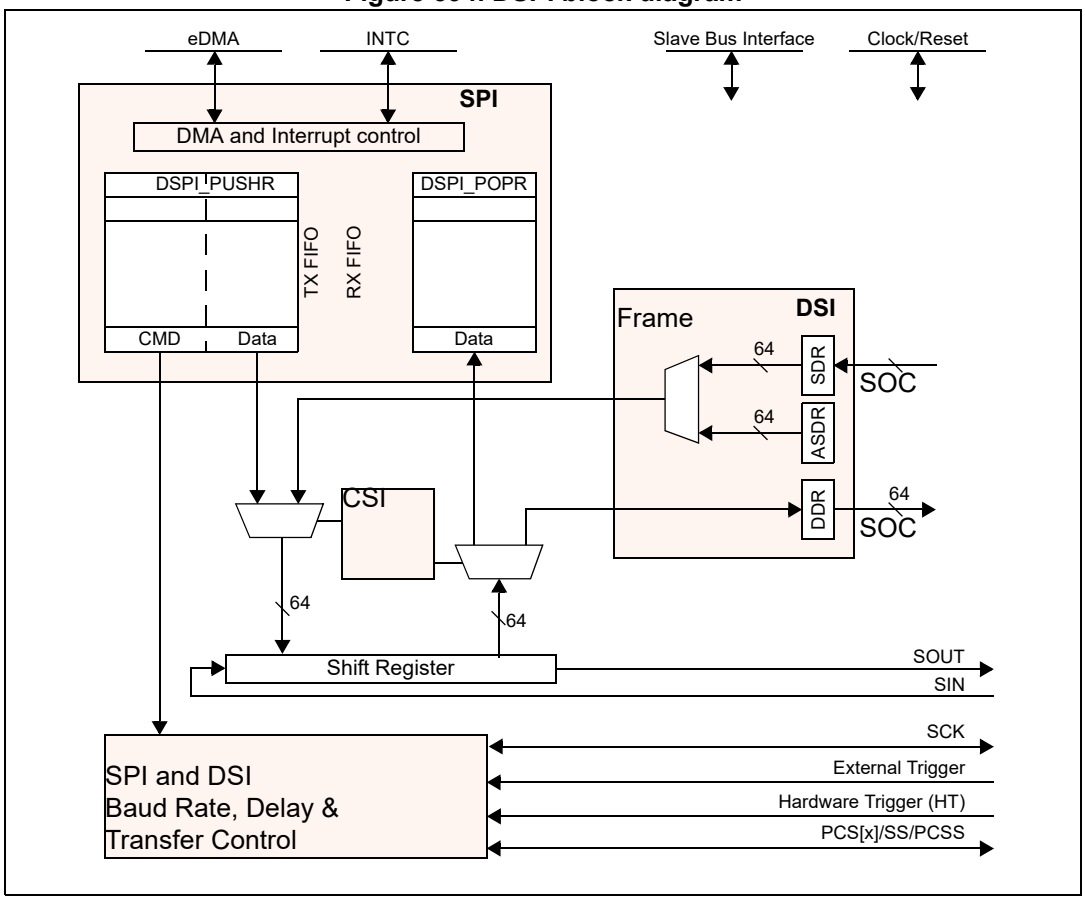
The Deserial Serial Peripheral Interface (DSPI) module provides a synchronous serial bus for communication between an MCU and an external peripheral device. The module supports MCU pin count reduction through serialization and deserialization of MCU internal signals transmitted over the SPI serial link.

Note: For the chip-specific implementation details of the instances of this module, see the Device Configuration chapter.

46.1.1 Block diagram

Figure 594 provides the block diagram of this module.

Figure 594. DSPI block diagram



46.1.2 Features

The DSPI supports the following features:

- Full-duplex, four-wire synchronous transfers.
- Master and Slave modes: data streaming operation in slave mode with continuous slave selection.
- Buffered transmit operation with 4-entry TX FIFO buffer.
- Buffered receive operation with 4-entry RX FIFO buffer.
- Asynchronous clocking scheme for Register and Protocol interfaces.
- Individual TX and RX FIFOs disabling for low-latency updates to SPI queues.
- Visibility inside TX and RX FIFOs for 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 are 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

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
  - bits in a memory-mapped register
- Interrupt conditions:
  - deserialized data matches pre-programmed pattern (DDIF)
  - transfer of current DSI frame complete (DSITCF)
  - DSI parity error (DPEF)
- Transfer initiation conditions:
  - continuous
  - change in data
- Pin serialization/deserialization with interleaved SPI frames for control and diagnostics.
- Continuous serial communications clock.

DSPI supports a combination of SPI and DSI modes of operation (Combined Serial Interface (CSI)) for the downstream Micro Second Channel in either Timed Serial Bus (TSB) configuration or Interleaved Frames Configuration (configurable by software). Both TSB and Interleaved TSB modes have the following common features:

- Transmission of frames is performed at frame boundaries.
- Frames from SPI and DSI are identifiable by a bit transmitted at the start of each frame.
- Separate interrupts for frame completion from SPI and DSI.

46.1.3 DSPI configurations

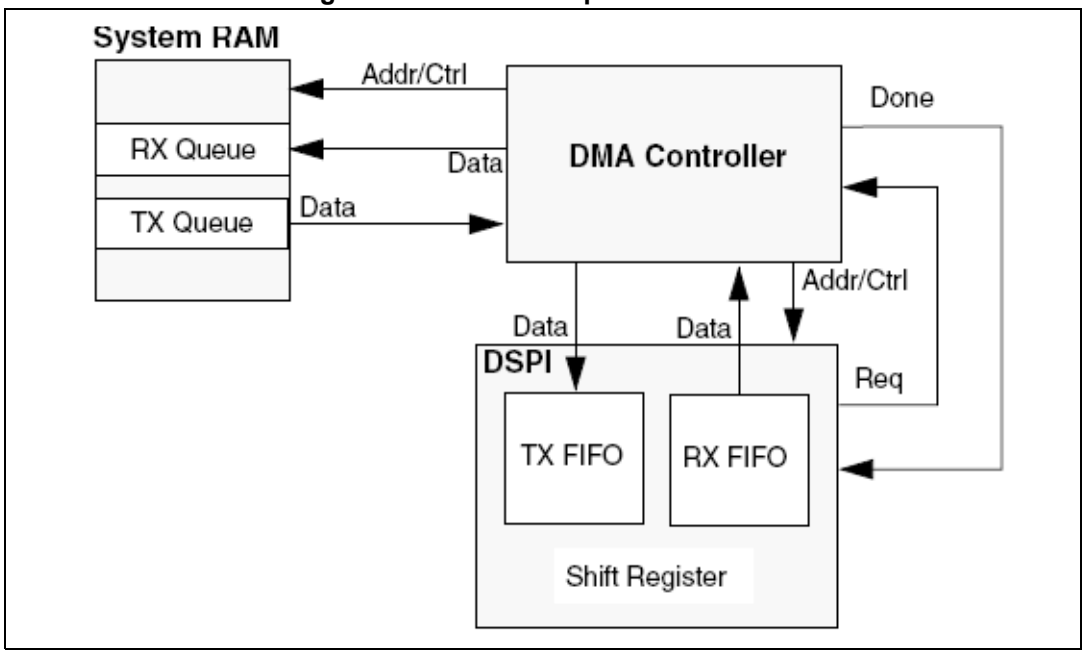
The DSPI module can operate in three configurations: SPI, DSI, and CSI. See the relevant section in the ‘Communication interfaces’ section of the ‘Device Configuration’ chapter.

46.1.3.1 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 595* shows a system example with DMA, DSPI and external queues in system RAM.

Figure 595. DSPI with queues and DMA



46.1.3.2 DSI configuration

In the DSI configuration the DSPI serializes up to 64 parallel input signals or register bits. The DSPI also deserializes the received data to parallel output signals or to a memory-mapped register. The data is transferred using a SPI-like protocol.

TSB mode provides the Micro Second downstream Channel support (MSC), serializing from 4 to 64 parallel input signals or register bits. See *Section 46.5.9: Timed Serial Bus (TSB)*.

46.1.3.3 CSI configuration

There are three configurations available in this mode.

- The normal CSI configuration is a combination of the SPI and DSI configurations. In this configuration the DSPI interleaves DSI data frames with SPI data frames. Interleaving is done on the frame boundaries.
- In the TSB configuration, transmission of SPI data has higher priority than DSI data.
- In the Interleaved TSB (ITSB) configuration, the frames from SPI and DSI are interleaved without priority. On every trigger, frames from DSI are sent when there are no frames in the SPI or the previous transmission was a frame from SPI. ITSB is detailed in *Section 46.5.10: Interleaved TSB (ITSB) Mode*.

46.1.4 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.

46.1.4.1 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.

46.1.4.2 Slave mode

Slave mode allows the DSPI to communicate with SPI/DSI bus masters 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.

46.1.4.3 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.

46.1.4.4 External Stop mode

External Stop mode is used for MCU power management; the DSPI supports the Peripheral Bus Stop mode mechanism.

Upon a request to enter External Stop mode:

- In Master 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.
- In Slave mode, a request to enter External Stop mode must be run after the last clock cycle or Low-power mode is not entered.

46.1.4.5 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.

46.2 DSPI signal description

This section provides the DSPI signals description.

Table 628 lists the signals that may connect off chip depending on device implementation.

Table 628. DSPI signal description

Signal	Description	I/O
PCS0/ SS	Master mode: Peripheral Chip Select 0 output Slave mode: Slave Select input	I/O
PCS[3:1]	Master mode: Peripheral Chip Select 1–3 Slave mode: Unused	O
PCS4	Master mode: Peripheral Chip Select 4	O
PCS5/ PCSS(1)	Master mode: Peripheral Chip Select 5 / Peripheral Chip Select Strobe Slave mode: Unused	O
PCS[7:6]	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

1. This pin is not available for DSPI_2

46.2.1 PCS0/$\overline{SS}$ — Peripheral Chip Select/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.

46.2.2 PCS1 – PCS3 — Peripheral Chip Selects 1 – 3

PCS1 – PCS3 are Peripheral Chip Select output signals in master mode.

In slave mode, these signals are unused.

46.2.3 PCS4 — Peripheral Chip Select 4

In master mode, PCS4 is a Peripheral Chip Select output signal.

46.2.4 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 $\overline{\text{PCSS}}$ signal acts as a strobe to an external peripheral chip select demultiplexer, which decodes the PCS0 – PCS4and PCS6 – PCS7 signals, preventing glitches on the demultiplexer outputs.

This signal is not used in slave mode.

46.2.5 PCS[6] – PCS[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.

46.2.6 SIN — Serial Input

SIN is a serial data input signal.

46.2.7 SOUT — Serial Output

SOUT is a serial data output signal.

46.2.8 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.

46.3 Memory map/register definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Write access to the DSPI_POPR also results in a transfer error.

Table 629. DSPI memory map

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_0000	DSPI Module Configuration Register (DSPI_MCR)	32	R/W	0000_4001h	*on page 1106*
0000_0008	DSPI Transfer Count Register (DSPI_TCR)	32	R/W	0000_0000h	*on page 1109*
0000_000C	DSPI Clock and Transfer Attributes Register 0 (In Master Mode) (DSPI_CTAR0)	32	R/W	7800_0000h	*on page 1110*
0000_000C	DSPI Clock and Transfer Attributes Register 0 (In Slave Mode) (DSPI_CTAR0_SLAVE)	32	R/W	7800_0000h	*on page 1115*
0000_0010	DSPI Clock and Transfer Attributes Register 1 (In Master Mode) (DSPI_CTAR1)	32	R/W	7800_0000h	*on page 1110*
0000_0010	DSPI Clock and Transfer Attributes Register 1 (In Slave Mode) (DSPI_CTAR1_SLAVE)	32	R/W	7800_0000h	*on page 1115*
0000_0014	DSPI Clock and Transfer Attributes Register 2 (In Master Mode) (DSPI_CTAR2)	32	R/W	7800_0000h	*on page 1110*
0000_0018	DSPI Clock and Transfer Attributes Register 3 (In Master Mode) (DSPI_CTAR3)	32	R/W	7800_0000h	*on page 1110*

Table 629. DSPI memory map(Continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_001C	DSPI Clock and Transfer Attributes Register 4 (In Master Mode) (DSPI_CTAR4)	32	R/W	7800_0000h	on page 1110
0000_0020	DSPI Clock and Transfer Attributes Register 5 (In Master Mode) (DSPI_CTAR5)	32	R/W	7800_0000h	on page 1110
0000_0024	DSPI Clock and Transfer Attributes Register 6 (In Master Mode) (DSPI_CTAR6)	32	R/W	7800_0000h	on page 1110
0000_0028	DSPI Clock and Transfer Attributes Register 7 (In Master Mode) (DSPI_CTAR7)	32	R/W	7800_0000h	on page 1110
0000_002C	DSPI Status Register (DSPI_SR)	32	R/W	0201_0000h	on page 1116
0000_0030	DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)	32	R/W	0000_0000h	on page 1119
0000_0034	DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR)	32	R/W	0000_0000h	on page 1122
0000_0034	DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)	32	R/W	0000_0000h	on page 1124
0000_0038	DSPI POP FIFO Register (DSPI_POPR)	32	R	0000_0000h	on page 1125
0000_003C	DSPI Transmit FIFO Register 0 (DSPI_TXFR0)	32	R	0000_0000h	on page 1126
0000_0040	DSPI Transmit FIFO Register 1 (DSPI_TXFR1)	32	R	0000_0000h	on page 1126
0000_0044	DSPI Transmit FIFO Register 2 (DSPI_TXFR2)	32	R	0000_0000h	on page 1126
0000_0048	DSPI Transmit FIFO Register 3 (DSPI_TXFR3)	32	R	0000_0000h	on page 1126
0000_007C	DSPI Receive FIFO Register 0 (DSPI_RXFR0)	32	R	0000_0000h	on page 1126
0000_0080	DSPI Receive FIFO Register 1 (DSPI_RXFR1)	32	R	0000_0000h	on page 1126
0000_0084	DSPI Receive FIFO Register 2 (DSPI_RXFR2)	32	R	0000_0000h	on page 1126
0000_0088	DSPI Receive FIFO Register 3 (DSPI_RXFR3)	32	R	0000_0000h	on page 1126
0000_00BC	DSPI DSI Configuration Register 0 (DSPI_DSICR0)	32	R/W	0000_0000h	on page 1127
0000_00C0	DSPI DSI Serialization Data Register 0 (DSPI_SDR0)	32	R	0000_0000h	on page 1129
0000_00C4	DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)	32	R/W	0000_0000h	on page 1130
0000_00C8	DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0)	32	R	0000_0000h	on page 1131

Table 629. DSPI memory map(Continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_00CC	DSPI DSI Deserialization Data Register 0 (DSPI_DDR0)	32	R	0000_0000h	on page 1131
0000_00D0	DSPI DSI Configuration Register 1 (DSPI_DSICR1)	32	R/W	0000_0000h	on page 1132
0000_00D4	DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0)	32	R/W	0000_0000h	on page 1133
0000_00E8	DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)	32	R/W	0000_0000h	on page 1134
0000_00EC	DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)	32	R/W	0000_0000h	on page 1135
0000_00F0	DSPI DSI Serialization Data Register 1 (DSPI_SDR1)	32	R	0000_0000h	on page 1135
0000_00F4	DSPI DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)	32	R/W	0000_0000h	on page 1136
0000_00F8	DSPI DSI Transmit Comparison Register 1 (DSPI_COMPR1)	32	R	0000_0000h	on page 1137
0000_00FC	DSPI DSI Deserialization Data Register 1 (DSPI_DDR1)	32	R	0000_0000h	on page 1137
0000_0100	DSPI DSI Serialization Source Select Register 1 (DSPI_SSR1)	32	R/W	0000_0000h	on page 1138
0000_0114	DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)	32	R/W	0000_0000h	on page 1139
0000_0118	DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)	32	R/W	0000_0000h	on page 1139
0000_011C	DSPI Clock and Transfer Attributes Register Extended 0 (DSPI_CTARE0)	32	R/W	0000_0001h	on page 1140
0000_0120	DSPI Clock and Transfer Attributes Register Extended 1 (DSPI_CTARE1)	32	R/W	0000_0001h	on page 1140
0000_0124	DSPI Clock and Transfer Attributes Register Extended 2 (DSPI_CTARE2)	32	R/W	0000_0001h	on page 1140
0000_0128	DSPI Clock and Transfer Attributes Register Extended 3 (DSPI_CTARE3)	32	R/W	0000_0001h	on page 1140
0000_012C	DSPI Clock and Transfer Attributes Register Extended 4 (DSPI_CTARE4)	32	R/W	0000_0001h	on page 1140
0000_0130	DSPI Clock and Transfer Attributes Register Extended 5 (DSPI_CTARE5)	32	R/W	0000_0001h	on page 1140
0000_0134	DSPI Clock and Transfer Attributes Register Extended 6 (DSPI_CTARE6)	32	R/W	0000_0001h	on page 1140

Table 629. DSPI memory map(Continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_0138	DSPI Clock and Transfer Attributes Register Extended 7 (DSPI_CTARE7)	32	R/W	0000_0001h	*on page 1140*
0000_013C	DSPI Status Register Extended (DSPI_SREX)	32	R	0000_0000h	*on page 1141*

46.3.1 DSPI Module Configuration Register (DSPI_MCR)

The DSPI_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.

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	0	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 596. DSPI Module Configuration Register (DSPI_MCR)

Table 630. DSPI_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 630. DSPI_MCR field descriptions(Continued)

Field	Description
2–3 DCONF	DSPI Configuration Selects the DSPI configuration. 00 SPI 01 DSI 10 CSI 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. MTFE should not be set in ITSB Mode as Microsecond Channel upstream operation is not supported. 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]/PCSS is used as the Peripheral Chip Select[5] signal (PCS[5]). 1 PCS[5]/PCSS is used as an active-low PCS Strobe signal (PCSS).
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.
8–15 PCSIS[7:0]	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx 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 PCSx is low. 1 The inactive state of PCSx is high.
16 Reserved	This read-only bitfield is reserved and always has the value zero.
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory-mapped logic in the DSPI, putting the DSPI in a software controlled power-saving state. The reset value of the MDIS bit is 1. MDIS should be set to ‘0’ in Slave Mode as a slave does not control master transactions. 0 Enable DSPI clocks. 1 Allow external logic to disable DSPI clocks.

Table 630. DSPI_MCR field descriptions(Continued)

Field	Description
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the DSPI (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 TX FIFO and 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	Clear RX FIFO Flushes 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 Enables use of CTARE (Command and Transfer Attribute Register Extended) Registers, which allow the user to send up to 32-bit SPI frames. Command Cycling is also enabled, allowing the user to send multiple Data Frames in a single Command Frame. When MCR[DIS_TXF] is asserted, Extended SPI Mode cannot be used to transmit SPI frames over 16 bits in size. 0 Normal SPI Mode. Up to 16-bit Frames. Command Cycling is not available. 1 Extended SPI Mode. Up to 32-bit SPI Frames. Command Cycling Enabled

Table 630. DSPI_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. The individual delay masks are selected via PUSHR[PE_MASC] and PUSHR{PP_MCSC]. The firmware should select appropriate masks when providing continuous frames via the PUSHR. This masking is not available if Continuous SCK mode is enabled. 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.

46.3.2 DSPI Transfer Count Register (DSPI_TCR)

DSPI_TCR has an SPI transfer counter to assist in queue management.

Do not write the TCR when the DSPI is in the Running state.

Address: Base + 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 597. DSPI Transfer Count Register (DSPI_TCR)

Table 631. DSPI_TCR field descriptions

Field	Description
0–15 SPI_TCNT	SPI Transfer Counter The SPI_TCNT field increments every time the last bit of a SPI frame is transmitted. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. A value written to SPI_TCNT presets the counter to that value. The Transfer Counter wraps around; incrementing past 65535 resets the counter to zero.
16–31 Reserved	This read-only bitfield is reserved and always has the value zero.

46.3.3 DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*)

CTARs are used to define different transfer attributes:

- 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 set the slave transfer attributes.

Do not write to the CTARs while the DSPI is in the Running state.

When the DSPI is configured as:

- SPI master – the CTAS field in the command portion of the TX FIFO entry selects which CTAR is used.
- SPI bus slave – the CTAR0 register is used.
- DSI master – the DSICTAS field in the DSPI DSI Configuration Register 0 (DSICR0) selects which CTAR is used.
- DSI bus slave – the CTAR1 register is used.

In CSI Configuration, the transfer attributes are based on whether the current frame is:

- SPI data – follow the protocol described for SPI Configuration,
- DSI data – follow the protocol described for DSI Configuration.

CSI Configuration is valid only in master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks via PDT and DT fields.

Address: Base + 0x000C 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 598. DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*)

Table 632. DSPI_CTAR*n* field descriptions

Field	Description
0 DBR	Double Baud Rate. (Master mode only) Doubles the baud rate of the Serial Communications Clock (SCK). It halves the Baud Rate division ratio, supporting faster frequencies and odd division ratios for the Serial Communications Clock (SCK). When DBR 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 633*. See the BR field description in this table for details on how to compute the baud rate. Refer to *Table 633* 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 value for the number of bits to be transmitted for any mode is 4.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) 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. When the DSPI operates in TSB mode, the FMSZ field value plus 1 is equal to the data frame bit number, where control of the PCS assertion switches from the DSICR0 to the DSICR1 register. If TSB Mode is operated when DSI is used in 64-bit mode, then DSICR0[FMSZ4] also comes into picture while switching PCS assertion control.
5 CPOL	Clock Polarity. (Master and Slave mode) Selects the inactive state of the Serial Communications Clock (SCK). Devices must have identical clock polarities for successful communication between serial devices. 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 or TSB mode. (Master and Slave mode) Selects which edge of SCK causes data to change and which edge causes data to be captured. Devices must have identical clock phase settings for successful communication between serial devices. In Continuous SCK or TSB modes, this bit 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.

Table 632. DSPI_CTAR*n* field descriptions(Continued)

Field	Description
7 LSBFE	LSB First. Specifies whether the LSB or MSB of the frame is transferred first. In TSB mode, this bit should comply with the MSC specification. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
8–9 PCSSCK	PCS to SCK Delay Prescaler. Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description in this table for calculating the PCS to SCK Delay. Refer to *Section 46.5.5.2: PCS to SCK delay (tCSC)* for more details. In TSB mode, PCSSCK has no effect. 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 in this table for calculating the After SCK Delay. Refer to *Section 46.5.5.3: After SCK delay (tASC)* for more details. In TSB mode, PASC has no effect. 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. (Master mode only) 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. In TSB mode, the PDT field defines two MSB bits of the Delay after Transfer. See the DT field description for calculating the Delay after Transfer. Refer to *Section 46.5.5.4: 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. (Master mode only) Selects the prescaler value for the baud rate. 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 in this table for calculating 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 632. DSPI_CTAR*n* field descriptions(Continued)

Field	Description
16–19 CSSCK	PCS to SCK Delay Scaler. (Master mode only) Selects the scaler value for the PCS to SCK delay. 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 calculated by: $t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$ *Table 634* lists the delay scaler values. Refer to *Section 46.5.5.2: PCS to SCK delay (tCSC)* for more details. In TSB mode, the field has no effect.
20–23 ASC	After SCK Delay Scaler. (Master mode only) Selects the scaler value for the After SCK Delay. 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 calculated by: $t_{ASC} = (1/f_P) \times PASC \times ASC$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer to *Section 46.5.5.3: After SCK delay (tASC)* for more details.
24–27 DT	Delay After Transfer Scaler (Master mode only) Selects the Delay after Transfer Scaler. 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 calculated by: $t_{DT} = (1/f_P) \times PDT \times DT$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. In the TSB mode, the Delay after Transfer is equal to a number formed by concatenation of the PDT and DT fields plus 1 of the SCK clock periods. Refer to *Section 46.5.5.2: PCS to SCK delay (tCSC)* for more details.
28–31 BR	Baud Rate Scaler. (Master mode only) Selects the scaler value for the baud rate. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is calculated by: $SCK\ baud\ rate = (f_P/PBR) \times [(1+DBR)/BR]$ *Table 635* lists the baud rate scaler values.

Table 633. 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

Table 633. DSPI SCK duty cycle(Continued)

DBR	CPHA	PBR	SCK duty cycle
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

Table 634. Delay scaler encoding

Field value	Delay scaler value
0000	2
0001	4
0010	8
0011	16
0100	32
0101	64
0110	128
0111	256
1000	512
1001	1024
1010	2048
1011	4096
1100	8192
1101	16384
1110	32768
1111	65536

Table 635. DSPI baud rate scaler

CTARn[BR]	Baud rate scaler value
0000	2
0001	4
0010	6
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256

Table 635. DSPI baud rate scaler(Continued)

CTARn[BR]	Baud rate scaler value
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

46.3.4 DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR*n*_SLAVE)

When the DSPI is configured as:

- SPI master – the CTAS field in the command portion of the TX FIFO entry selects which CTAR is used.
- SPI bus slave – the CTAR0 register is used.
- DSI master – the DSICTAS field in the DSPI DSI Configuration Register 0 (DSICR0) selects which CTAR is used.
- DSI bus slave – the CTAR1 register is used.

In CSI Configuration, the transfer attributes are based on whether the current frame is:

- SPI data – follow the protocol described for SPI Configuration,
- DSI data – follow the protocol described for DSI Configuration.

CSI Configuration is valid only in master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks via PDT and DT fields.

Address: Base + 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	FMSZ5	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 599. DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR*n*_SLAVE)

Table 636. DSPI_CTAR*n*_SLAVE field descriptions

Field	Description
0–4 FMSZ	Frame Size The number of bits transferred per frame is equal to the concatenated field {CTARn_SLAVE[FMSZ5], CTARn_SLAVE[FMSZ]} value plus 1. The minimum valid FMSZ field value is 4.
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 or TSB mode. (Master and Slave mode) Selects which edge of SCK causes data to change and which edge causes data to be captured. Devices must have identical clock phase settings for successful communication between serial devices. In Continuous SCK or TSB modes, 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 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 FMSZ5	MSB of Frame Size when DSI is used in 64-bit Mode The number of bits transferred per frame is equal to the concatenated field {CTARn_SLAVE[FMSZ5], CTARn_SLAVE[FMSZ]} value plus 1.
10–31 Reserved	This read-only bitfield is reserved and always has the value zero.

46.3.5 DSPI Status Register (DSPI_SR)

The status register contains DSPI status bits and interrupt/DMA request event flag bits.

Software can clear most of the flag bits in the SR by writing a ‘1’ to them (w1c). 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.

Address: Base + 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	SPITCF	EOQF	TFUF	DSITCF	TFFF	BSYF	CMDTCF	DPEF	SPEF	DDIF	RFOF	TFIWF	RFDF	CMDFFF
W	w1c		w1c	w1c	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 600. DSPI Status Register (DSPI_SR)

Table 637. DSPI_SR field descriptions

Field	Description
0 TCF	Transfer Complete Flag. Indicates that all bits in a frame have been shifted out. 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 SPITCF	SPI Frame Transfer Complete Flag. Similar to TCF but only asserted on SPI frame transmission completion in CSI Mode This includes ITSB and 'TSB in CSI' modes, but not SPI or DSI Modes. 0 SPI frame transfer is not complete. 1 SPI frame transfer is complete.
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. 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.
4 TFUF	Transmit FIFO Underflow Flag. Indicates an underflow condition in the TX FIFO. Transmit underflow is detected only for DSPI blocks 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. 0 No Tx FIFO underflow. 1 Tx FIFO underflow has occurred.

Table 637. DSPI_SR field descriptions(Continued)

Field	Description
5 DSITCF	DSI Frame Transfer Complete Flag. Similar to TCF but only asserted on DSI frame transmission completion in CSI Mode. This includes ITSB and 'TSB in CSI' modes, but not SPI or DSI Modes. 0 DSI frame transfer is not complete. 1 DSI frame transfer is complete.
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. See *Section 46.5.2.5: Command First In First Out (CMD FIFO) buffering mechanism*. This bit is valid only when DSPI_MCR[XSPI] is enabled. 0No 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 DPEF	DSI Parity Error Flag. Indicates that a DSI frame with parity error has been received. 0 No parity error. 1 Parity error has occurred.
10 SPEF	SPI Parity Error Flag. Indicates that a SPI frame with parity error has been received. 0 No parity error. 1 Parity error has occurred.
11 DDIF	DSI Data Received with Active Bits. Indicates that a DSI frame has been received with bits selected by DIMR with active polarity, defined by the DPIR. 0 No DSI data with active bits was received. 1 DSI data with active bits was received.

Table 637. DSPI_SR field descriptions(Continued)

Field	Description
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. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
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.

46.3.6 DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)

The DSPI_RSER controls DMA and interrupt requests.

Do not write to the RSER while the DSPI is in the Running state.

Address: Base + 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF_RE	CMDFFF_RE	SPITCF_RE	EOQF_RE	TFUF_RE	DSITCF_RE	TFFF_RE	TFFF_DIRS	CMDTCF_RE	DPEF_RE	SPEF_RE	DDIF_RE	RFOF_RE	TFIWF_RE	RFDF_RE	RFDF_DIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMDFFF_DIRS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 601. DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)

Table 638. DSPI_RSER field descriptions

Field	Description
0 TCF_RE	Transmission Complete Request Enable. Enables TCF flag in the Status register to generate an interrupt request. 0 Disabled. 1 Enabled.
1 CMDFFF_RE	Command FIFO Fill Flag Request Enable. Enables the CMDFFF flag in the Status register to generate a request. DSPI_RSER[CMDFFF_DIRS] toggles either an interrupt or a DMA request. 0 Disabled. 1 Enabled.
2 SPITCF_RE	SPI Frame Transmission Complete Request Enable. Enables SPITCF flag in the Status register to generate an interrupt request. 0 Disabled. 1 Enabled.
3 EOQF_RE	DSPI Finished Request Enable Enables the EOQF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
4 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
5 DSITCF_RE	DSI Frame Transmission Complete Request Enable. Enables DSITCF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.

Table 638. DSPI_RSER field descriptions(Continued)

Field	Description
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the status register to generate a request. DSPI_RSER[TFFF_DIRS] toggles between either an interrupt or a DMA request. 0 Disabled. 1 Enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. DSPI_SR[TFFF] flag and DSPI_RSER[TFFF_RE] must be set. 0 Interrupt requests. 1 DMA requests.
8 CMDTCF_RE	Command Transmission Complete Request Enable. The CMDTCF_RE bit enables CMDTCF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
9 DPEF_RE	DSI Parity Error Request Enable Enables the DPEF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
10 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
11 DDIF_RE	DSI data received with active bits Request Enable Enables the DDIF flag in the status register to generate an interrupt requests. 0 Disabled. 1 Enabled.
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
13 TFIWF_RE	Transmit FIFO Invalid Write Request Enable Enables the TFIWF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
14 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the status register to generate a request. DSPI_RSER[RFDF_DIRS] toggles either an interrupt or a DMA request. 0 Disabled 1 Enabled
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. DSPI_SR[RFDF] flag and DSPI_RSER[RFDF_RE] must be set. 0 Interrupt request. 1 DMA request.

Table 638. DSPI_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. DSPI_SR[CMDFFF] flag and DSPI_RSER[CMDFFF_RE] must be set. 0 Interrupt request. 1 DMA request.
17–31 Reserved	This read-only bitfield is reserved and always has the value zero.

46.3.7 DSPI PUSH FIFO Register In Master Mode (DSPI_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 for 8- or 16-bit writes to the Data field of PUSHR.
- The CMD FIFO for writes to the Command field of PUSHR.

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 the 16-bit Command Field of PUSHR is reserved.

When Extended SPI Mode is not enabled (MCR[XSPI] = 0):

- TX FIFO and CMD FIFO must be filled simultaneously.
- Writes must be given to both Data and Command fields of PUSHR for every PUSHR operation.
- TX FIFO and CMD FIFO can be considered a single 32-bit FIFO.

When Extended SPI Mode is enabled (MCR[XSPI] = 1):

- 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 do not update the FIFO. Reads during Module disable mode return the last PUSHR write performed when Module was enabled.

Address: Base + 0x0034 Access: User read/write

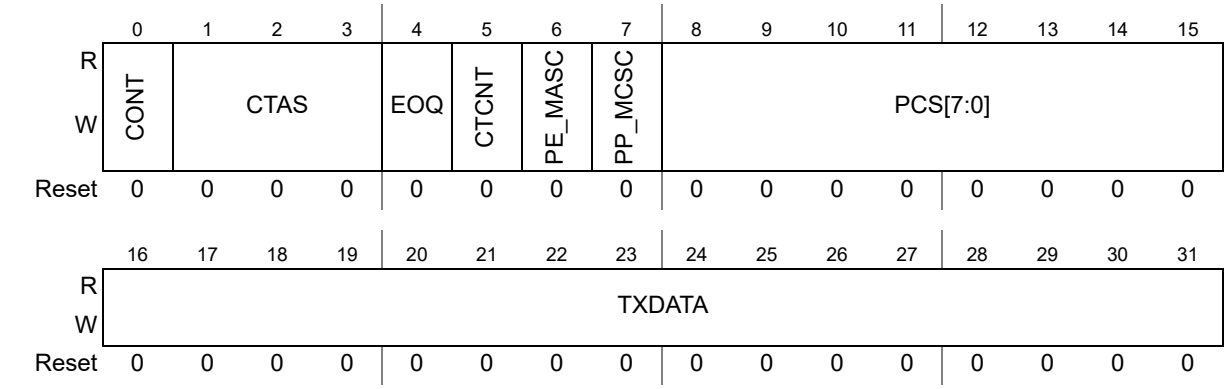


Figure 602. DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR)

Table 639. DSPI_PUSHR field descriptions

Field	Description
0 CONT	Continuous Peripheral Chip Select Enable. (SPI master mode) Continuous Selection Format enables the selected PCS signals to remain asserted between transfers. PUSHR[EOQ] and PUSHR[CONT] bits cannot be asserted together in any Command Frame as 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 the Device Configuration chapter to determine how many CTARs this device has. Do 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. DSPI_SR[EOQF] is set at the end of the transfer. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.

Table 639. DSPI_PUSHR field descriptions(Continued)

Field	Description
5 CTCNT	Clear Transfer Counter. Clears the SPI_TCNT field in the Transfer Count register. The SPI_TCNT field is cleared before the DSPI starts transmitting the current SPI frame. Hence the current SPI frame causes 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 46.5.6.6: Fast Continuous Selection Format* for more details. 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: t_{ASC} 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: t_{ASC} delay is masked in the current frame.
7 PP_MCSC	Parity Polarity or Mask t_{CSC} delay in the next frame PP: 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 46.5.6.6: Fast Continuous Selection Format* for more details. 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: t_{CSC} 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: t_{CSC} delay is masked in the next frame.
8–15 PCS[7:0]	Select which PCS signals are to be asserted for the transfer. Refer to the chip configuration chapter for the number of PCS signals used in this MCU. 0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command

46.3.8 DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)

PUSHR provides the means to write to the TX FIFO.

Data written to this register is transferred to:

- The TX FIFO for 8- or 16-bit writes to the Data field of PUSHR.

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 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.

Address: Base + 0x0034 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 603. DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)

Table 640. DSPI_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.

46.3.9 DSPI POP FIFO Register (DSPI_POPR)

POPR is used to read the RX FIFO.

8- or 16-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register generates a Transfer Error.

Address: Base + 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 604. DSPI POP FIFO Register (DSPI_POPR)

Table 641. DSPI_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.

46.3.10 DSPI Transmit FIFO Registers (DSPI_TXFRn)

TXFRn provide visibility into the TX FIFO for debugging purposes.

Each read-only register is an entry in the TX FIFO. Reading the TXFRn registers does not alter the state of the TX FIFO.

Reading DSPI_TXFRn registers is invalid for a DSPI Master when used in Extended SPI (XSPI) mode.

Address: Base + 0x003C 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 605. DSPI Transmit FIFO Registers (DSPI_TXFRn)

Table 642. DSPI_TXFRn field descriptions

Field	Description
0–15 TXCMD_TXDATA	Transmit Command or Transmit Data In master mode, the TXCMD field 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.

46.3.11 DSPI Receive FIFO Registers (DSPI_RXFRn)

RXFRn provide visibility into the RX FIFO for debugging purposes.

Each read-only register is an entry in the RX FIFO. Reading the RXFRn registers does not alter the state of the RX FIFO.

MCR[MDIS] must be 0 when RXFR is read.

Address: Base + 0x007C 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 606. DSPI Receive FIFO Registers (DSPI_RXFRn)

Table 643. DSPI_RXFRn field descriptions

Field	Description
0–31 RXDATA	Receive Data Contains the received SPI data.

46.3.12 DSPI DSI Configuration Register 0 (DSPI_DSICR0)

DSICR0 selects various attributes associated with DSI and CSI Configurations.

Do not write to the DSICR0 while the DSPI is in the Running state.

Address: Base + 0x00BC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved	FMSZ4	Reserved						FMSZ5	Reserved	ITSB	TSBC	TXSS	TPOL(1)	Reserved	CID
Reset	0	0	0	0	0	0	0	0	0	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	DCONT	DSICTAS[2:0]			DMS	PES	PE	PP	DPCSx							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 607. DSPI DSI Configuration Register 0 (DSPI_DSICR0)

1. This bit is reserved if the device does not have a hardware trigger input signal (HT).

Table 644. DSPI_DSICR0 field descriptions

Field	Description
1 FMSZ4	MSB of the frame size in master mode when DSI is used in 32-bit mode. If the bit is set, 16 is added to the frame size, as defined by the CTARn[FMSZ] field. The CTARx register is selected by the DSICTAS field.
8 FMSZ5	MSB of the frame size in master mode when DSI is used in 64-bit mode. If the bit is set, 32 is added to the frame size, as defined by the concatenation of {DSICR0[FMSZ4], CTARn[FMSZ]} fields. The CTARx register is selected by the DSICTAS field. Hence the final concatenated frame size using this bit is {DSICR0[FMSZ5], DSICR0[FMSZ4], CTARn[FMSZ]}. This field is only valid when DSICR1[DSI64] is enabled.
10 ITSB	Interleaved TSB mode. Enables the Interleaved TSB mode of operation. When enabled, there is no priority among frames from the SPI/DSI. DSI frames are sent when either the previous transmission was an SPI frame or the TX_FIFO is empty. Frames are transmitted on every trigger whose source is selected by the TRG bit setting. ITSB mode requires DSICR0[TSBC] set and should be written only when MCR[HALT] is asserted. If ITSB bit is set without setting DSICR0[TSBC], DSPI operates in Normal Mode depending on MCR[DCONF] bit. See *Section 46.5.10: Interleaved TSB (ITSB) Mode* for more details. 0 Disabled. 1 Enabled.
11 TSBC	Timed Serial Bus Configuration. Enables the Timed Serial Bus Configuration, which allows up to 64-bit data to be used. It also allows t_{DT} to be programmable. 0 Disabled. 1 Enabled.
12 TXSS	Transmit Data Source Select. Selects the source of data to be serialized. The source can be either: a) data from host software written to the DSPI DSI Alternate Serialization Data Register (ASDR1/0), or b) parallel input pin states latched into the DSPI DSI Serialization Data Register (SDR1/0). 0 SDR source. 1 ASDR source.
15 CID	Change In Data Transfer Enable When the CID bit is set, DSI frames are initiated when the current DSI data differs from the previous DSI data shifted out. Do not enable CID when DCONT is enabled. 0 Disabled. 1 Enabled.
16 DCONT	DSI Continuous Peripheral Chip Select Enable (DSI Master mode only) Enables the PCS signals to remain asserted between transfers. When the TSBC bit is set, DCONT has no effect. Do not enable DCONT when CID is enabled. 0 PCS signals return to inactive. 1 PCS signals remain asserted.

Table 644. DSPI_DSICR0 field descriptions(Continued)

Field	Description
17–19 DSICTAS[2:0]	DSI Clock and Transfer Attributes Select (DSI Master mode only) Selects the CTAR which provides transfer attributes for DSI frames. In DSI slave mode, CTAR1 is always selected.
20 DMS	Data Match Stop Stops DSI frame transmissions if DDIF flag is set in the status register. 0 Disabled. 1 Enabled.
21 PES	Parity Error Stop Stops DSI operation if a parity error occurred in the received DSI frame. 0 Disabled. 1 Enabled.
22 PE	Parity Enable Enables parity bit transmission and parity reception check for the DSI frames. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame.
23 PP	Parity Polarity Controls the polarity of the parity bit transmitted and checked. 0 Even Parity: the number of ‘1’ bits in the transmitted frame is even. The SR[DPEF] bit is set if in the received frame number of ‘1’ bits is odd. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[DPEF] bit is set if in the received frame number of 1 bits is even.
24–31 DPCSx	DSI Peripheral Chip Select 0–7 Selects which of the PCS signals to assert during a DSI master mode transfer. 0 Negate PCS[x]. 1 Assert PCS[x].

46.3.13 DSPI DSI Serialization Data Register 0 (DSPI_SDR0)

Read-only SDR0 contains the states of the 32 LSB parallel input signals.

The states of these signals are latched into the SDR0 on the rising edge of every protocol clock.

When the TXSS bit in the DSICR0 is cleared, the data in the SDR0 is used as the source of the DSI frames.

When DSICR1[DSI64E] is set, SDR1 is the MSB half and SDR0 is the LSB half of the 64-bit SDR.

Address: Base + 0x00C0 Access: User read-only

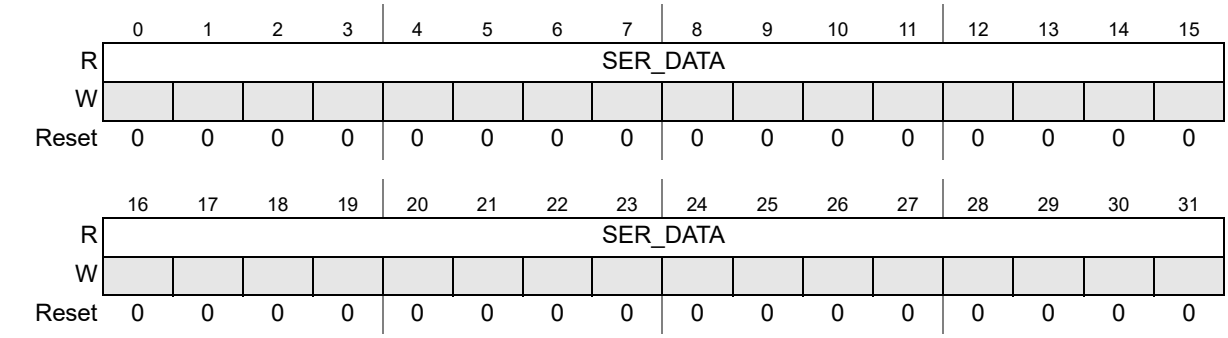


Figure 608. DSPI DSI Serialization Data Register 0 (DSPI_SDR0)

Table 645. DSPI_SDR0 field descriptions

Field	Description
0–31 SER_DATA	Serialized Data Contains the signal states of the 32 LSB parallel input signals.

46.3.14 DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)

ASDR0 is used by host software to write the 32 LSB of the data to be serialized.

When the TXSS bit in the DSICR0 is set, the data in the ASDR0 is the source of the DSI frames.

Writes to the ASDR0 take effect on the next frame boundary.

When DSICR1[DSI64E] is set, ASDR1 is the MSB half and ASDR0 is the LSB half of the 64-bit ASDR.

Address: Base + 0x00C4 Access: User read/write

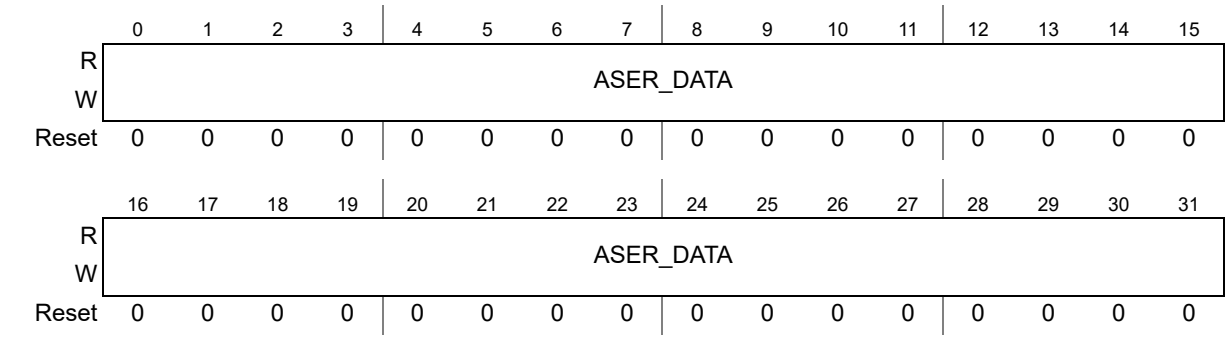


Figure 609. DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)

Table 646. DSPI_ASDR0 field descriptions

Field	Description
0–31 ASER_DATA	Alternate Serialized Data Holds the alternate 32 LSB of data to be serialized.

46.3.15 DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0)

Read-only COMPR0 holds a copy of the 32 LSB of last transmitted DSI data.

DSI data is transferred to this register as it is loaded into the TX Shift Register.

When DSICR1[DSI64E] is set, COMPR1 is the MSB half and COMPR0 is the LSB half of the 64-bit COMPR.

Address: Base + 0x00C8 Access: User read-only

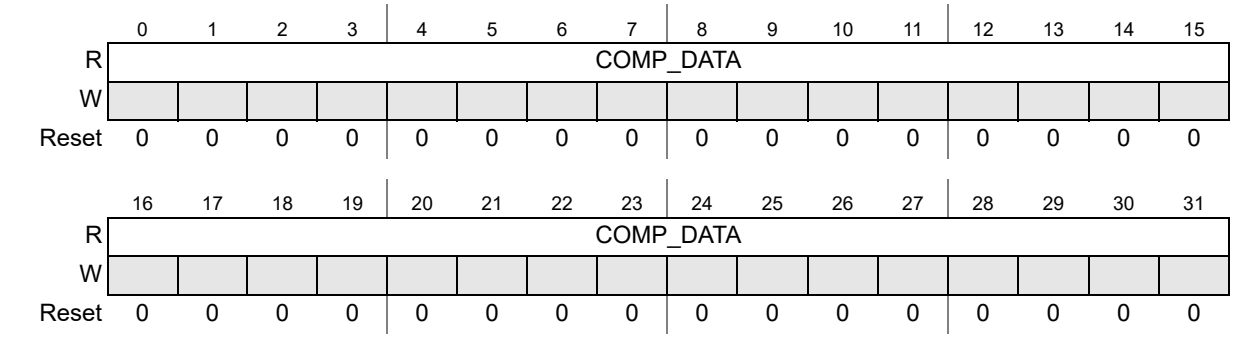


Figure 610. DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0)

Table 647. DSPI_COMPR0 field descriptions

Field	Description
0–31 COMP_DATA	Compare Data Holds the 32 LSB of last serialized DSI data.

46.3.16 DSPI DSI Deserialization Data Register 0 (DSPI_DDR0)

Read-only DDR0 holds the 32 LSB signal states for the Parallel Output signals.

Host software can read data from incoming DSI frames.

When DSICR1[DSI64E] is set, DDR1 is the MSB half and DDR0 is the LSB half of the 64-bit DDR.

Address: Base + 0x00CC Access: User read-only

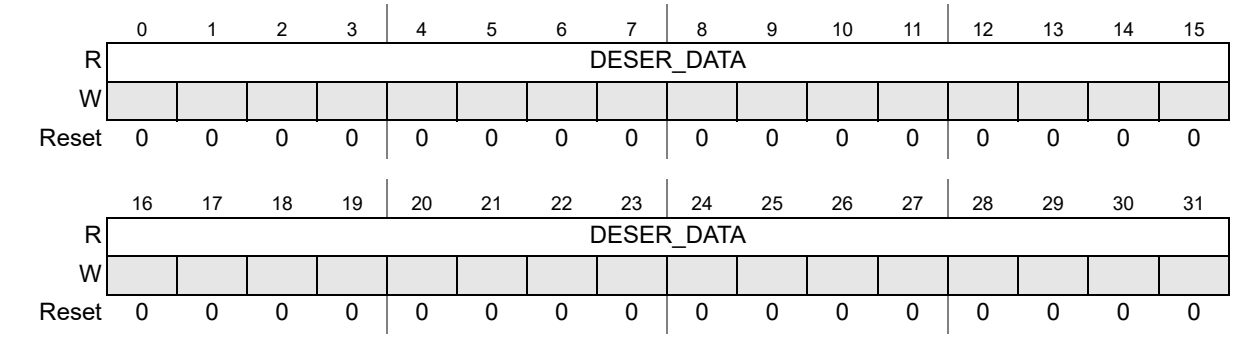


Figure 611. DSPI DSI Deserialization Data Register 0 (DSPI_DDR0)

Table 648. DSPI_DDR0 field descriptions

Field	Description
0–31 DESER_DATA	Deserialized Data Holds 32 LSB of deserialized data presented as signal states to the Parallel Output signals.

46.3.17 DSPI DSI Configuration Register 1 (DSPI_DSICR1)

DSICR1 selects various attributes associated with TSB (and ITSB) Configuration.

Do not write to the DSICR1 while the DSPI is in the Running state.

Address: Base + 0x00D0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	TSBCNT						0	0	0	0	0	DSI64E	DSE 1	DSE 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRGPRD								DPCS1_x							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 612. DSPI DSI Configuration Register 1 (DSPI_DSICR1)

Table 649. DSPI_DSICR1 field descriptions

Field	Description
0-1 Reserved	This read-only bitfield is reserved and always has the value zero.
2-7 TSBCNT	Timed Serial Bus Operation Count TSBCNT defines the length of the data frame When TSBC is set. When DSI 64-bit mode is enabled (DSI64E), the length value can vary from 3 to 63 (000011b to 111111b); otherwise, it can vary from 3 to 31 (000011b to 011111b). The TSBCNT field selects the number of data bits to be shifted out during a transfer in TSB mode. Number of data bits in data frame = TSBCNT + 1.
8–12 Reserved	This read-only bitfield is reserved and always has the value zero.
13 DSI64E	DSI 64-bit Mode Enable Allows serialization and deserialization of Data frames up to 64 bits in size. The Extended Registers DSPI_SDR1, DSPI_ASDR1, DSPI_COMPR1, DSPI_DDR1, DSPI_SSR1, DSPI_DIMR1, DSPI_DPIR1 are valid in this Mode. 0 Disabled. DSI Mode operates in the default 32-bit configuration 1 Enabled. DSI Mode operates in 64-bit configuration

Table 649. DSPI_DSICR1 field descriptions(Continued)

Field	Description
14 DSE1	Data Select Enable 1 When DSICR0[TBSC] bit is set, the DSE1 bit enables insertion of the zero bit (Data Select) in the middle of the data frame. The insertion bit position is defined by the FMSZ field of CTARn register, selected by the DSICR0[DSICTAS] field. If DSICR1[DSI64] is enabled, then the insertion bit position is defined by concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]} fields. The TSBCNT field value must be greater than the FMSZ field value plus one for proper operation of the DSE1 bit. 0 No Zero bit is inserted in the middle of the data frame. 1 Zero bit is inserted at the middle of the data frame. Total number of bits in the data frame is increased by 1.
15 DSE0	Data Select Enable 0 When DSICR0[TSBC] bit is set, the DSE0 bit controls insertion of the zero bit (Data Select) in the beginning of the Data frame. 0 No Zero bit is inserted in the beginning of the frame. 1 Zero bit is inserted at the beginning of the data frame. Total number of bits in the data frame is increased by 1.
16–23 TRGPRD	Internal Trigger Period for the ITSB mode. When DSICR0[TRG] bit is cleared and ITSB Mode is enabled, this field determines the trigger period for the internal trigger in number of baud rate clock cycles. The trigger period should be calculated as: Trigger Period = 64 bits (DSI frame maximum size) + 2 selection bits (maximum) + t_{PF} (passive frame time) When DSICR1[DSI64E] is disabled, DSI frame maximum size is 32. Refer CTAR[FMSZ] field for minimum frame size allowed. See Interleaved TSB (ITSB) Mode for more details. 0 – 4 These values are invalid since minimum Command frame size is 5. (i.e. 1 selection bit + minimum frame size which is 4) 5 – 255 Number of baud rate clock cycles to be used as the trigger period. The trigger period implemented is equal to TRGPRD + 1.
24–31 DPCS1_x	DSI Peripheral Chip Select 0–7 These bits define the PCSs to assert for the second part of the DSI frame when operating in TSB (and ITSB) configuration with dual receiver. The DPCS1 bits select which of the PCS signals to assert during the second part of the DSI frame. The DPCS1 bits control the assertions of the PCS signals only in TSB and ITSB mode. 0 Negate PCS[x]. 1 Assert PCS[x].

46.3.18 DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0)

SSR0 is used to create a combined frame for transmission that contains bits from ASDR0 and from SDR0.

Each bit in the SSR0 register selects a corresponding bit to be serialized.

When DSICR0[TXSS] is set, the SSR0 register value has no effect.

When DSICR1[DSI64E] is set, SSR1 is the MSB half and SSR0 is the LSB half of the 64-bit SSR.

Address: Base + 0x00D4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SS															
Reset	0	0	0	0	0	0	0	0	0	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	SS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 613. DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0)

Table 650. DSPI_SSR0 field descriptions

Field	Description
0–31 SS	Source Select Select the serialization source for the 32 LSB of DSI frame. Each SS bit selects data for a corresponded bit in the transmitted frame. 0The bit in the transmitted frame is taken from the parallel input pin. 1The bit in the transmitted frame is taken from the ASDR0 register

46.3.19 DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)

DIMR0 selects bits in the 32 LSB of received DSI frame to be checked to generate the DDI interrupt.

When DSICR1[DSI64E] is set, DIMR1 is the MSB half and DIMR0 is the LSB half of the 64-bit DIMR.

Address: Base + 0x00E8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MASK															
Reset	0	0	0	0	0	0	0	0	0	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	MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 614. DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)

Table 651. DSPI_DIMR0 field descriptions

Field	Description
0–31 MASK	Mask 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame produces a DDI interrupt if the data bit matches the configured polarity.

46.3.20 DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)

DPIR0 defines which data bit value in the 32 LSB of received DSI frame generates the DDI interrupt.

When DSICR1[DSI64E] is set, DPIR1 is the MSB half and DPIR0 is the LSB half of the 64-bit DPIR.

Address: Base + 0x00EC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DP															
Reset	0	0	0	0	0	0	0	0	0	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	DP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 615. DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)

Table 652. DSPI_DPIR0 field descriptions

Field	Description
0–31 DP	Data Polarity 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

46.3.21 DSPI DSI Serialization Data Register 1 (DSPI_SDR1)

Read-only SDR1 contains the states of the 32 MSB parallel input signals.

The states of these signals are latched into the SDR1 on the rising edge of every protocol clock.

When DSICR0[TXSS] is cleared, the data in SDR1 and SDR0 is the source of the DSI frames.

The concatenation of {DSPI_SDR1, DSPI_SDR0} provides the 64-bit data to be serialized.

This register is valid only when DSICR1[DSI64E] is set.

Address: Base + 0x00F0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 616. DSPI DSI Serialization Data Register 1 (DSPI_SDR1)

Table 653. DSPI_SDR1 field descriptions

Field	Description
0–31 SER_DATA	Serialized Data

46.3.22 DSPI DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)

ASDR1 is used by host software to write the 32 MSB of the 64-bit data to be serialized.

When DSICR0[TXSS] is set, the data in ASDR1 and ASDR0 is the source of the DSI frames.

Writes to the ASDR1 take effect on the next frame boundary.

The concatenation of {DSPI_ASDR1, DSPI_ASDR0} provides the 64-bit alternate serialization data.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: Base + 0x00F4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ASER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ASER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 617. DSPI DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)

Table 654. DSPI_ASDR1 field descriptions

Field	Description
0–31 ASER_DATA	Alternate Serialized Data

46.3.23 DSPI DSI Transmit Comparison Register 1 (DSPI_COMPR1)

Read-only COMPR1 holds a copy of the 32 MSB of the last transmitted 64-bit DSI data.

The upper 32 bits of the 64-bit DSI data is transferred to this register as it is loaded into the TX Shift Register.

The concatenation of {DSPI_COMPR1, DSPI_COMPR0} provides the 64-bit transmit comparison data.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: Base + 0x00F8 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	COMP_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 618. DSPI DSI Transmit Comparison Register 1 (DSPI_COMPR1)

Table 655. DSPI_COMPR1 field descriptions

Field	Description
0–31 COMP_DATA	Compare Data

46.3.24 DSPI DSI Deserialization Data Register 1 (DSPI_DDR1)

Read-only DDR1 holds the 32 MSB signal states for the 64 Parallel Output signals.

Host software can read data from incoming DSI frames.

The concatenation of {DSPI_DDR1, DSPI_DDR0} provides the 64-bit Deserialized data.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: Base + 0x00FC Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DESER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DESER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 619. DSPI DSI Deserialization Data Register 1 (DSPI_DDR1)

Table 656. DSPI_DDR1 field descriptions

Field	Description
0–31 DESER_DATA	Deserialized Data

46.3.25 DSPI DSI Serialization Source Select Register 1 (DSPI_SSR1)

SSR1 is used to create a combined frame for transmission that contains bits from ASDR1 and from SDR1.

Each bit in the SSR1 register selects a corresponding bit to be serialized.

When DSICR0[TXSS] is set, the SSR1 register value has no effect.

The concatenation of {DSPI_SSR1, DSPI_SSR0} provides the 64-bit Serialization Source Select Value.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: Base + 0x0100 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 620. DSPI DSI Serialization Source Select Register (DSPI_SSR)

Table 657. DSPI_SSR field descriptions

Field	Description
0–31 SS	Source Select 0The bit in the transmitted frame is taken from the parallel input pin. 1The bit in the transmitted frame is taken from the ASDR1 register

46.3.26 DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)

DIMR1 selects bits from the 32 MSB in the received 64-bit DSI frame to be checked to generate the DDI interrupt.

The concatenation of {DSPI_DIMR1, DSPI_DIMR0} provides the 64-bit Deserialized Data Interrupt Mask Value.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: Base + 0x0114 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MASK															
Reset	0	0	0	0	0	0	0	0	0	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	MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 621. DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)

Table 658. DSPI_DIMR1 field descriptions

Field	Description
0–31 MASK	Mask 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame can produce a DDI interrupt if the data bit matches the configured polarity.

46.3.27 DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)

DPIR1 defines which of the 32 MSB value in the received 64-bit DSI frame generates the DDI interrupt.

The concatenation of {DSPI_DPIR1, DSPI_DPIR0} provides the 64-bit Deserialized Data Polarity Value.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: Base + 0x0118 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DP															
Reset	0	0	0	0	0	0	0	0	0	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	DP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 622. DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)

Table 659. DSPI_DPIR1 field descriptions

Field	Description
0–31 DP	Data Polarity 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

46.3.28 DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*)

CTARE registers are used to define the extended transfer attributes for an SPI frame.

These registers are valid only when DSPI_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.
- an SPI bus slave, the CTARE0 register is used.

Address: Base + 0x011C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	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 623. DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*)

Table 660. DSPI_CTARE*n* field descriptions

Field	Description
16 FMSZE	Frame Size Extended This field concatenated with DSPI_CTAR[FMSZ] defines the frame size of the SPI frames to be transmitted. Frame size is the concatenation of {DSPI_CTAR[FMSZ], DSPI_CTARE[FMSZE]} + 1. 0Default Mode. Up to 16-bit SPI frames can be transferred. 1Up to 32-bit SPI frames can be transferred. Each frame transfer is a result of 2 TX FIFO Pops.
21–31 DTCP	Data Transfer Count Preload 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_CTARE register. The value 0 is reserved and should not be written in this field. The default value of this field is 1.

46.3.29 DSPI Status Register Extended (DSPI_SREX)

The DSPI_SREX contains status fields that reflect the DSPI status and indicate the occurrence of events.

This register is not writable.

Address: Base + 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 624. DSPI Status Register Extended (DSPI_SREX)

Table 661. DSPI_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.

46.4 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_MCR)
- DSPI Transfer Count Register (DSPI_TCR)
- DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTARn)
- DSPI Clock and Transfer Attributes Extended Register (In Master Mode) (DSPI_CTAREn)
- DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)
- DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR) [0:15]
- DSPI DSI Configuration Register 0 (DSPI_DSICR0)
- DSPI DSI Configuration Register 1 (DSPI_DSICR1)

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_SR) [0:15] (for stuck at 0)
- DSPI Status Register Extended (DSPI_SREX) [0:8] (for stuck at 0)
- DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)
- DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)
- DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)
- DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)

Non-Safety: All other registers and bits fall in this category.

46.5 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.

The DSPI has the following configurations:

- SPI Configuration in which the DSPI operates as a basic SPI or a queued SPI.
- DSI Configuration in which the DSPI serializes and deserializes Parallel Input/Output signals or bits from memory-mapped register.
- CSI Configuration in which the DSPI combines the functionality of the SPI and DSI configurations.

The DCONF field in the DSPI Module Configuration Register (MCR) determines the DSPI Configuration. See *Section 46.3.1: DSPI Module Configuration Register (DSPI_MCR)* for the DSPI configuration values.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows CTAR selection on a frame by frame basis by setting a field in the SPI command. Extended SPI

Mode (DSPI_MCR[XSPI]) further allows the usage of CTAREn (CTARn Extended) registers to send multiple Data frames (of up to 32-bit frame size) using a single Command frame.

DSI configuration statically selects which CTAR to use. In CSI, Configuration priority logic determines if SPI data or DSI data is transferred and dictates which CTAR is used for the data transfer.

See DSPI Clock and Transfer Attributes Registers for information on the fields of the CTARs.

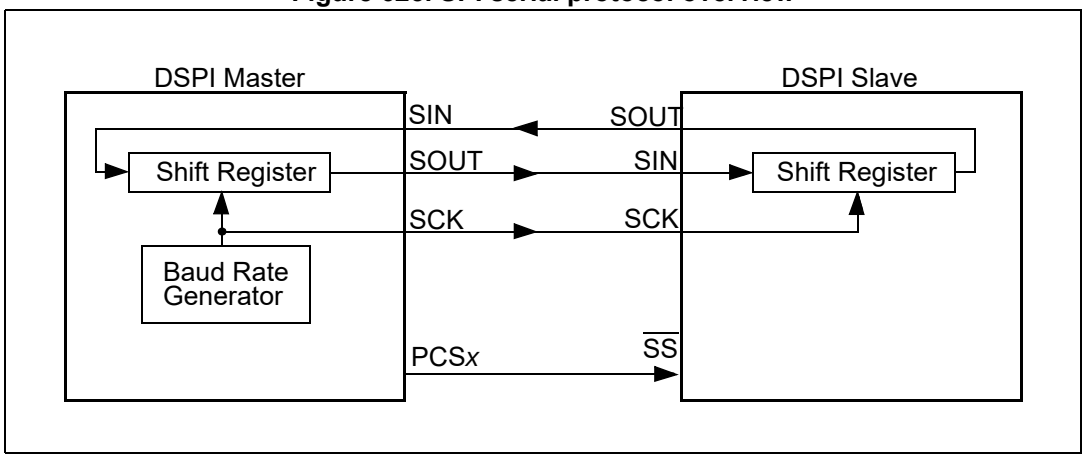
See DSPI Clock and Transfer Attributes Registers Extended for information on the fields of the CTARE registers.

Typical master to slave connections are shown in *Figure 625*.

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 Status Register is set to indicate a completed transfer.

Figure 625. 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 the chip configuration chapter for the number of PCS signals used in this MCU.

The three DSPI configurations share transfer protocol and timing properties which are described independently of the configuration in *Section 46.5.6: Transfer formats*.

The transfer rate and delay settings are described in *Section 46.5.5: DSPI baud rate and clock delay generation*.

46.5.1 Start and Stop of DSPI transfers

The DSPI has two operating states that are independent of DSPI configuration:

- STOPPED is the default DSPI 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.

- RUNNING is the state when serial transfers take place.

DSPI_SR[TXRXS] indicates what state the DSPI in ('1' = 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.

46.5.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The SPI frames can be 32 bits long.

The DSPI is in SPI Configuration when the DCONF field in the MCR is 0b00.

The host CPU or a DMA controller transfers the SPI data from memory external to DSPI RAM queues to a transmit FIFO (TX FIFO) buffer.

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 46.5.2.4: Transmit First In First Out (TX FIFO) buffering mechanism*, *Section 46.5.2.5: Command First In First Out (CMD FIFO) buffering mechanism* and *Section 46.5.2.6: Receive First In First Out (RX FIFO) Buffering Mechanism*. The interrupt and DMA request conditions are described in *Section 46.5.12: Interrupts/DMA requests*.

The SPI Configuration supports two block-specific modes:

- Master mode: the DSPI initiates and controls the transfer according to the fields of the executing CMD FIFO entry.
- 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.

46.5.2.1 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 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. Extended SPI Mode allows for larger frame sizes of up to 32 bits.

46.5.2.2 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 in CTAR0 and CTARE0 for successful communication with an SPI master. The data is shifted out with MSB first; shifting out of LSB is not supported in this mode.

46.5.2.3 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:

- MCR[DIS_TXF] disables TX FIFO and CMD FIFO.
  - TFFF, TFUF, CMDFFF, CMDCTR and TXCTR fields in the status register and the extended status register behave as if there is a one-entry FIFO.
  - The contents of the TXFR registers and TXNXTPTR and CMDNXTPTR are undefined.
- MCR[DIS_RXF] bit disables the RX FIFO.
  - The RFDF, RFOF and RXCTR fields in the status register behave as if there is a one-entry FIFO.
  - The contents of the RXFR registers and POPNXTPTR are undefined.

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.

46.5.2.4 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 that is added to the TX FIFO by writing to the Data field of DSPI PUSH FIFO Register (PUSHR).

The number of entries in the TX FIFO is device-specific.

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. 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. This 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.

The TXFRn Registers are invalid in Extended SPI Mode as TX FIFO and CMD FIFO are used separately.

46.5.2.4.1 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.When TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete, the TFFF bit is cleared. Writing a ‘1’ to the TFFF bit also clears it. 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.

The TFFF can generate a DMA request or an interrupt request. See *Section 46.5.12.2: Transmit FIFO Fill interrupt or DMA request* for details.

46.5.2.4.2 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_MCR[XSPI]) is enabled and the frame size of SPI Data to be transmitted is more than 16 bits, then two Data entries are popped from TX FIFO simultaneously and 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 46.5.12.9: Transmit FIFO Underflow interrupt request* for details.

46.5.2.5 Command First In First Out (CMD FIFO) buffering mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission.

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_MCR[XSPI]) is disabled:

- The TX FIFO and CMD FIFO must be filled together as every CMDFIFO entry has a corresponding single TXFIFO entry attached to it.

When Extended SPI Mode (DSPI_MCR[XSPI]) is enabled:

- The TX FIFO and CMD FIFO can be filled independently as every CMDFIFO entry can have multiple TXFIFO entries attached to it.
- The CTARE[DTCP] field decides the number of SPI Data Frames of size {FMSZE, FMSZ} to be transmitted using the current Command Entry. FMSZ and FMSZE fields are given in the CTAR/CTARE registers respectively, pointed by the CTAS field in the Command frame.
- The amount of time a command entry is in use is known as a Command Cycle. The Busy flag DSPI_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 is used during the next command cycle. This 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.

The TXFRn Registers are invalid in Extended SPI Mode as TX FIFO and CMD FIFO are used separately.

46.5.2.6 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 on 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 status register 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.

46.5.2.6.1 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 status register is set to indicating an overflow condition. Consequently:

- If the DSPI_MCR[ROOE] bit is set, the incoming data is shifted in to the shift register.
- If the DSPI_MCR[ROOE] bit is cleared, the incoming data is ignored.

46.5.2.6.2 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 status register 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.

46.5.3 Deserial Serial Interface (DSI) configuration

The DSI Configuration supports pin count reduction by serializing parallel input signals or register bits and shifting them out in a SPI-like protocol. The timing and transfer protocol is described in *Section 46.5.6: Transfer formats*. The received serial frames are converted to a parallel form (deserialized) and placed on the Parallel Output signals or in the DSPI_DDR. The various features of the DSI Configuration are set in DSPI DSI Configuration Registers (DSPI_DSICR1/0).

The DSI frames can be from 4 to 32 bits, or 64 bits when DSICR1[DSI64E] is enabled.

46.5.3.1 DSI master mode

In DSI master mode the DSPI initiates and controls the DSI transfers. The DSI master has four different conditions (described in *Section 46.5.3.5: DSI transfer initiation control*) that can initiate a transfer:

- Continuous.
- Change in data.
- Trigger signal.
- Trigger signal combined with a change in data.

Transfer attributes are set during initialization. DSICR0[DSICTAS] field determines which of the DSPI_CTARs will control the transfer attributes.

46.5.3.2 Slave Mode

In DSI slave mode the DSPI responds to transfers initiated by a SPI or DSI bus master. In this mode the DSPI does not initiate DSI transfers.

Certain transfer attributes such as clock polarity and phase must be set for successful communication with a DSI master. The DSI slave mode transfer attributes are set in the DSPI_CTAR1.

The data is shifted out with MSB first.

46.5.3.3 DSI serialization

The DSI configuration from 4 to 64 bits can be serialized using two different sources. The TXSS bit in the DSPI_DSICR0 selects between the DSPI DSI Serialization Data Register (DSPI_SDR0-1) and the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR0-1) as the source of the serialized data. The DSPI_SDR0 holds the latest 32 parallel input signal values which is sampled at every rising edge of the Bus clock.

When DSICR1[DSI64E] is enabled, 64-bit DSI Frames are supported and hence DSPI_SDR1 holds the 32 MSB parallel input signal values which is sampled at every rising edge of the bus clock. The DSPI_ASDR0-1 registers are written by host software and used as an alternate source of serialized data. DSPI_ASDR1 is only usable when DSICR1[DSI64E] is enabled.

The DSPI has to sample the (SDR/ASDR) inputs before it transmits a DSI frame. Due to the asynchronous clocking structure within DSPI, the sampled data (from ASDR1/0 or SDR1/0) must be synchronized into the Protocol Domain before being loaded into the Shift Register for transmission.

To prevent Clock Domain clocking issues, SDR/ASDR inputs should not change while they are being sampled into the Protocol Domain. This is assured in the following manner:

- For the ASDR0 and ADSR1 inputs:
  - The initial write performed on ASDR register causes the 32-bit ASDR input to be latched into a temporary register which is then locked (further latching is prevented) until this data gets synchronized into the Protocol Domain.
  - Once synchronized, the lock is removed and the next ASDR write triggers this process again. If a write is performed on ASDR while the lock is asserted, this is processed as soon as the previous synchronization process completes and the lock is de-asserted. If multiple writes are performed on ASDR when lock is asserted after the first write, only the last value is considered for synchronization once the lock is de-asserted.
- For the SDR0 and SDR1 inputs:
  - The process is the same as ASDR with the only difference being that SDR inputs are delayed internally by a single clock to detect a change in the SDR input pins.

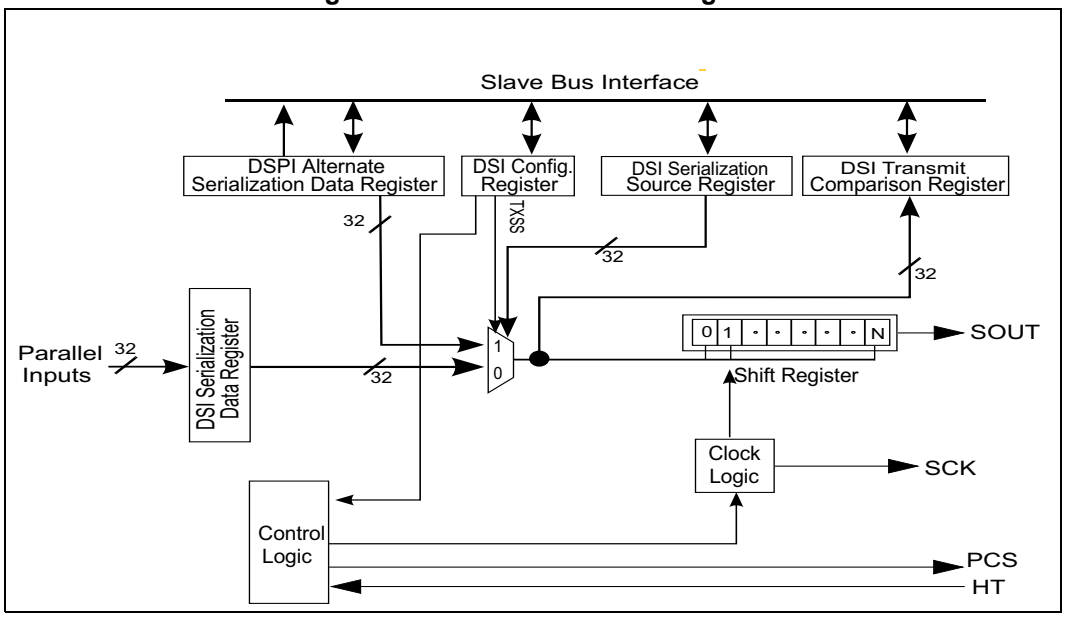
Once the synchronization process for a particular Deserial input begins, it takes up to five Register clock periods and five Protocol clock periods before the synchronization process for the next Deserial input can begin. Hence the writes performed on ASDR0/1 and SDR0/1 are constrained with this timing.

The parallel input pin state to be transmitted should be selected by TXSS or DSPI_SSR1/0 registers and the frame size should be higher than the bit position in the preselected frame.

DSPI_SSR0/1 registers provide additional way of creating the frame for transmission. Each bit from this register is ORed with the TXSS bit and controls individual transmitted bit source. This way, the transmitted frame can have any combination of the DSPI_SDR0/1 and DSPI_ASDR0/1 bits. This feature allows control SPI based devices, requiring control and data fields in the frame. The control field may come from DSPI_ASDR0/1 registers set by the CPU, while data field can be generated by device peripheral modules, like PWM timers.

A copy of the last DSI frame shifted out of the Shift Register is stored in the DSPI DSI Transmit Comparison Registers (DSPI_COMPR0/1). This register provides added visibility for debugging and it serves as a reference for transfer initiation control. *Figure 626* shows the DSI serialization logic.

Figure 626. DSI serialization diagram

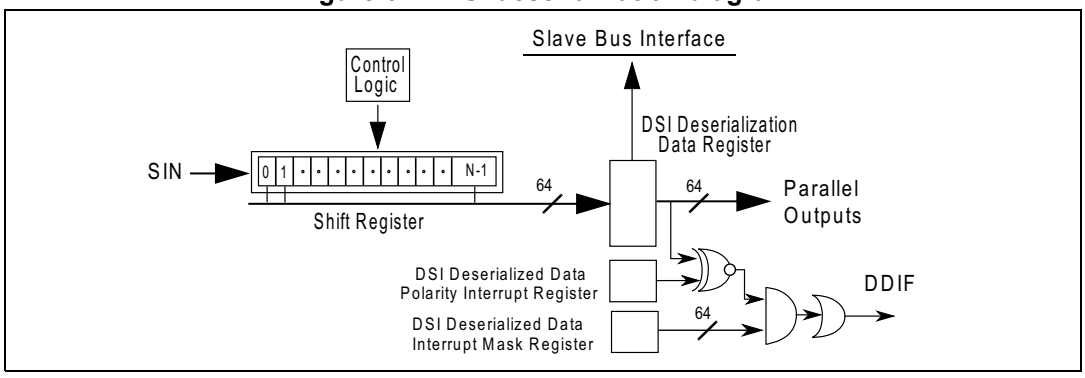


46.5.3.4 DSI deserialization

When all bits in a DSI frame have been shifted in, the frame is copied to the DSPI DSI Deserialization Data Register (DSPI_DDR0/1). This register presents the deserialized data as Parallel Output signal values. The DSPI_DDR0/1 is memory-mapped to allow host software to read the deserialized data directly.

The received data is bit-wise compared to the value of the DSI Deserialized Data Polarity Interrupt Register (DSPI_DPIR0/1), bit-wise ANDed with DSI Deserialized Interrupt Mask Register (DSPI_DIMR1/0) and the results ORed to produce DDIF flag in the DSPI_SR. Which in turn can cause DDIF interrupt request based on RSER[DDIF_RE] and/or stop DSI frame transmissions if DSICR0[DMS] bit is set. *Figure 627* shows the DSI Deserialization logic.

Figure 627. DSI deserialization diagram



46.5.3.5 DSI transfer initiation control

Data transfers for a master DSPI in DSI configuration are initiated by a condition. The transfer initiation conditions are selected by the CID bits in the DSPI_DSICR0 register. *Table 662* lists the four transfer initiation conditions.

Table 662. DSI data transfer initiation control

DSPI_DSICR0 bits	Transfer initiation control
CID	
0	Continuous
1	Change in Data
0	Triggered
1	Triggered or Change in Data

46.5.3.5.1 Continuous control

For continuous control, a new DSI frame shifts out when the previous transfer cycle has completed and the delay after transfer (t_{DT}) has elapsed.

46.5.3.5.2 Change in data control

For change in data control, a transfer is initiated when the data to be serialized has changed since the transfer of the last DSI frame. A copy of the previously transferred DSI data is stored in the DSPI_COMPR0/1. When the data selected for the transfer from the DSPI_SDR0/1 and DSPI_ASDR0/1 registers is different from the data in the DSPI_COMPR0/1 a new DSI frame is transmitted.

46.5.3.5.3 Change in data control

For change in data control, initiation of a transfer is controlled by the detection of a change in data to be serialized.

It is not advisable to configure a DSPI Slave with the combination DSICR0[DCONT] = 1, DSICR0[CID] = 1; since the following statement then holds true:

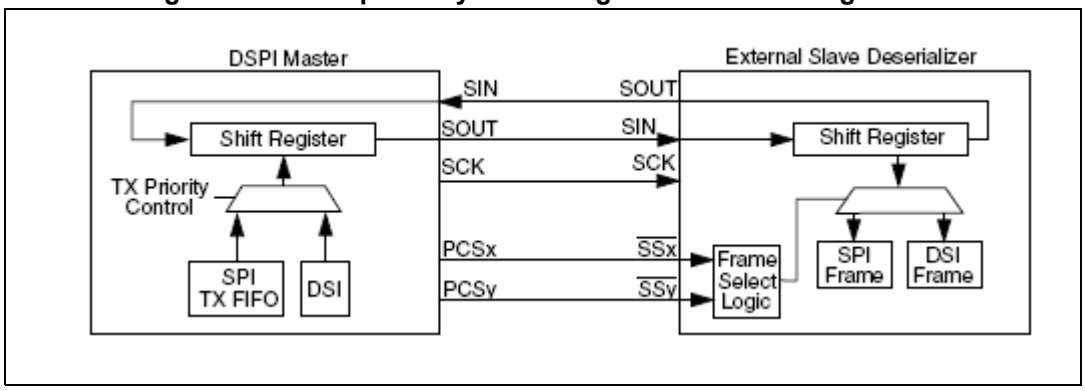
> Once slave select is asserted, there is a very small window (of up to three module clocks) after every SOF (Start of Frame) during which if a write on ASDR/SDR registers occurs, the next frame to be transmitted would be loaded into the shift register. Since it is not feasible that such a write is initiated and completed within this short span of time, the data transmitted by DSPI slave would lag by one data frame.

46.5.4 Combined Serial Interface (CSI) configuration

The CSI Configuration of the DSPI is used to support SPI and DSI functions on a frame by frame basis.

CSI Configuration allows interleaving of DSI data frames from the parallel input signals with SPI data frames from the TX FIFO. The data returned from the bus slave is either used to drive the Parallel Output signals or it is stored in the RX FIFO. The CSI Configuration allows serialized data and configuration or diagnostic data to be transferred to a slave device using only one serial link. The DSPI is in CSI Configuration when the DCONF field in the DSPI_MCR is 0b10. *Figure 628* shows an example of how a DSPI can be used with a deserializing peripheral that supports SPI control for control and diagnostic frames.

Figure 628. Example of System using DSPI in CSI Configuration



In CSI Configuration, the DSPI transfers DSI data based on *Section 46.5.3.5: DSI transfer initiation control*. When there are SPI commands in the TX FIFO, the SPI data has priority over the DSI frames. When the TX FIFO or CMD FIFO is empty, DSI transfer resumes.

Two peripheral chip select signals indicate whether DSI data or SPI data is transmitted. The user must configure the DSPI so that the two CTARs associated with DSI data and SPI data assert different peripheral chip select signals denoted in the figure as PCSx and PCSy.

The CSI Configuration is only supported in master mode.

Data returned from the external slave while a DSI frame is transferred is placed on the Parallel Output signals. Data returned from the external slave while a SPI frame is transferred is moved to the RX FIFO. The TX FIFO, CMD FIFO and RX FIFO are fully functional in CSI mode.

46.5.4.1 CSI serialization

Serialization in the CSI configuration is similar to serialization in DSI Configuration. The transfer attributes for SPI frames are determined by the DSPI_CTAR selected by the CTAS field in the SPI command halfword. The transfer attributes for the DSI frames are determined by the DSPI_CTAR selected by the DSICR0[DSICTAS] field.

The parallel inputs signal states are latched into the DSPI DSI Serialization Data Register (DSPI_SDR) on the rising edge of every protocol clock and serialized based on the transfer initiation control settings in the DSPI_DSICR0/1. When SPI frames are written to the TX FIFO (and when CMD FIFO is not empty), they have priority over DSI data and are transferred at the next frame boundary. A copy of the most recently transferred DSI frame is stored in the DSPI_COMPR0/1. The Transfer Priority Logic selects the source of the serialized data and asserts the appropriate PCS signal.

Separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI frames.

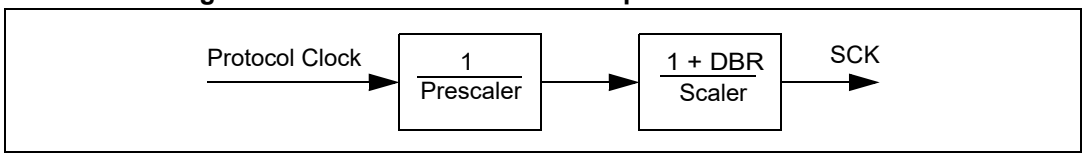
46.5.4.2 CSI deserialization

The deserialized frames in CSI Configuration goes into the DSPI_DDR1/0 or the RX FIFO based on the transfer priority logic. When DSI frames are transferred, the returned frames are deserialized and latched into the DSPI_DDR0/1. When SPI frames are transferred, the returned frames are deserialized and written to the RX FIFO.

46.5.5 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 629* shows conceptually how the SCK signal is generated.

Figure 629. Communications clock prescalers and scalers



46.5.5.1 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 663* shows an example of how to compute the baud rate.

Table 663. 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 663 are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

46.5.5.2 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 664* for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTARx registers select the PCS to SCK delay by the formula in the CSSCK field description. *Table 664* shows an example of how to compute the PCS to SCK delay.

Table 664. 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

PCSCSK and CSSCK fields have no effect in TSB configuration.

Note: The clock frequency mentioned in Table 664 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

46.5.5.3 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 665* for an illustration of the After SCK delay. The PASC and ASC fields in the CTARx registers select the After SCK Delay by the formula in the ASC field description.

Table 665 shows an example of how to compute the After SCK delay.

Table 665. After SCK delay computation example

f_P	PASC	Prescaler	ASC	Scaler	After SCK delay
100 MHz	0b01	3	0b0100	32	0.96 µs

PCASC and ASC fields have no effect in TSB configuration.

Note: The clock frequency mentioned in Table 665 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

46.5.5.4 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 666* for an illustration of the delay after transfer. the PDT and DT fields in the CTARx registers select the Delay after Transfer by the formula in the DT field description. *Table 666* shows an example of how to compute the delay after transfer.

Table 666. 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 666 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

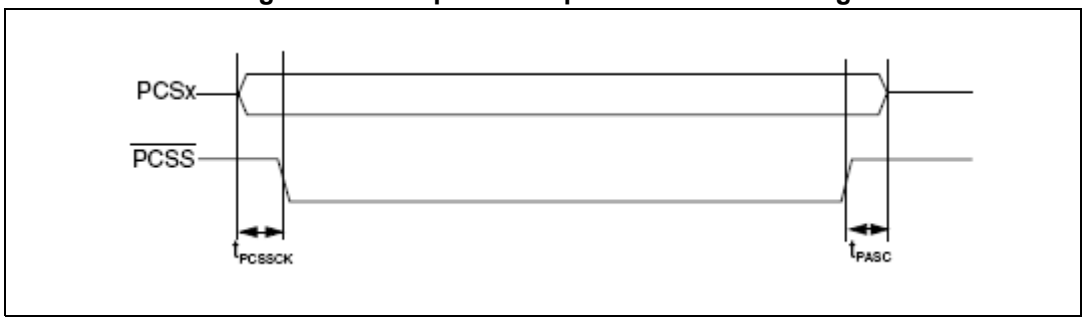
When in non-continuous clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] bitfield description. When in continuous clock mode and TSB is not enabled, the delay is fixed at 1 SCK period.

In TSB mode, the Delay after Transfer is equal to a number formed by concatenation of PDT and DT fields plus 1 of the SCK clock periods. See detailed information on *Section 46.5.9: Timed Serial Bus (TSB)*.

46.5.5.5 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 630* shows the timing of the $\overline{PCSS}$ signal relative to PCS signals.

Figure 630. 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 23

$$t_{PCSSCK} = \frac{1}{f_P} \times PCSSCK$$

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 24

$$t_{PASC} = \frac{1}{f_P} \times PASC$$

Table 667 shows an example of how to compute the t_{PCSSCK} delay.

Table 667. Peripheral Chip Select Strobe Assert Computation Example

f_P	PCSSCK	Prescaler	Delay before transfer
100 MHz	0b11	7	70.0 ns

Table 668 shows an example of how to compute the t_{PASC} delay.

Table 668. 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 or TSB mode are enabled.

Note: The clock frequency mentioned in the preceding Table 667 and Table 668 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

46.5.6 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 (CTARn) 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.

In DSI slave mode, only CTAR1 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.

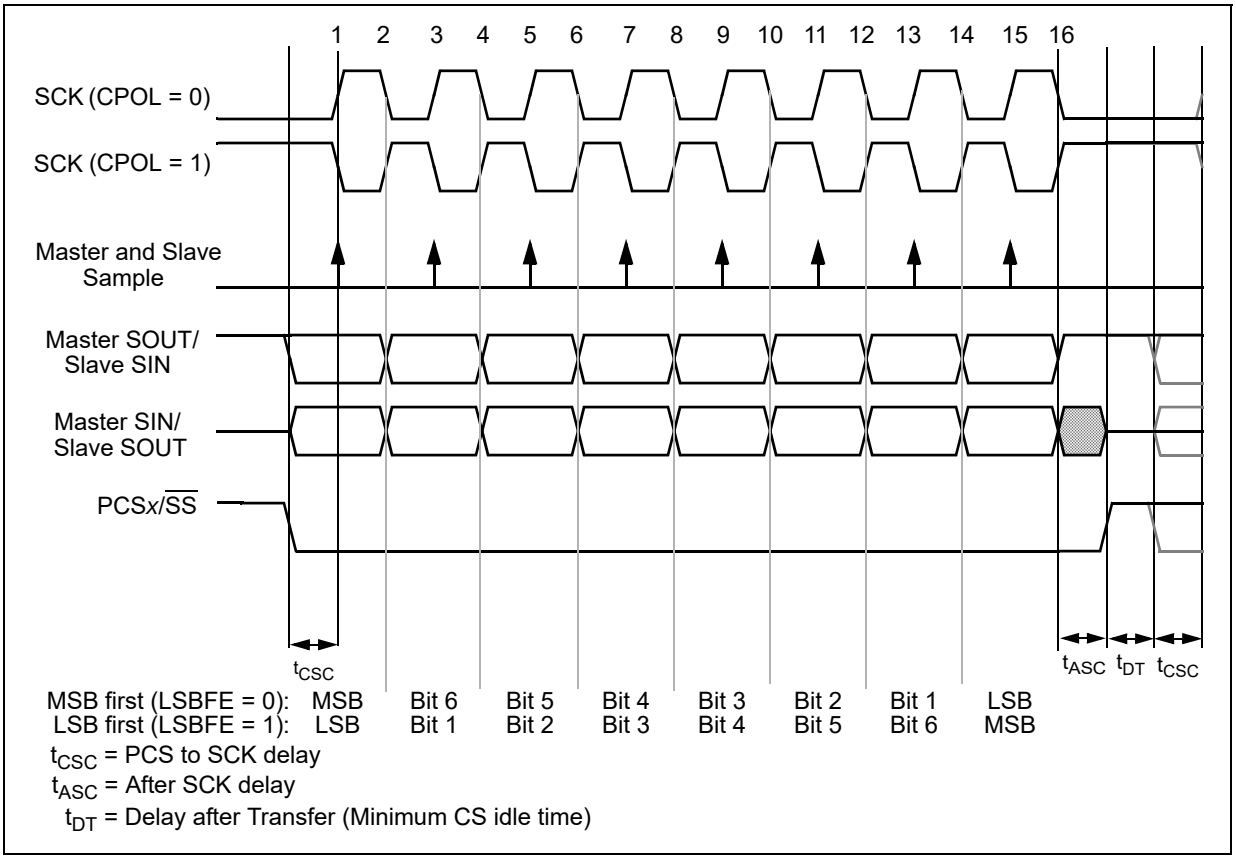
A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the SPI and DSI Configurations, the DSPI provides the option of keeping the PCS signals asserted between frames. See *Section 46.5.6.5: Continuous Selection Format* for details.

46.5.6.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in *Figure 631* 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 631. 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.

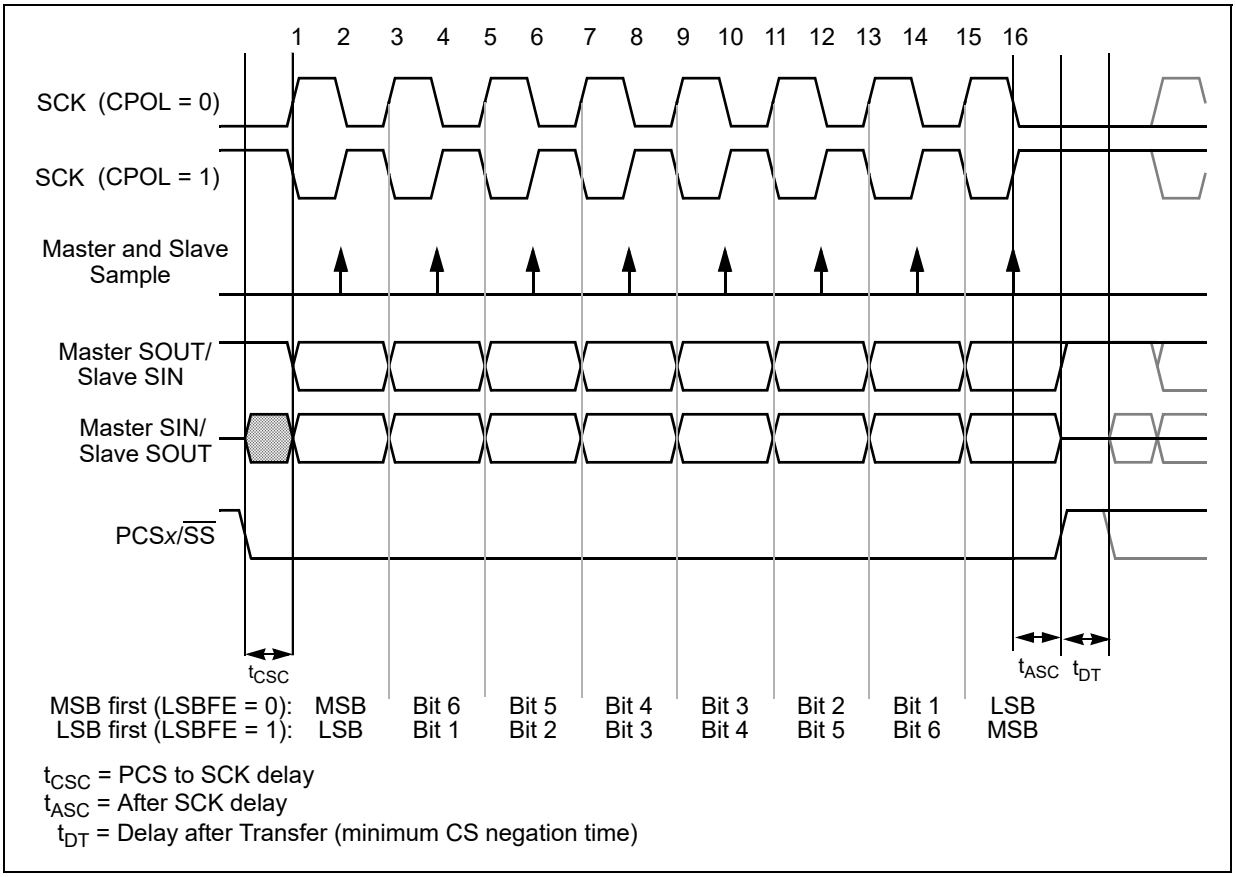
For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges.

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.

46.5.6.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in *Figure 632* 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 632. 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.

46.5.6.3 Modified SPI/DSI 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 633 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 to the 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 633. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/4$)

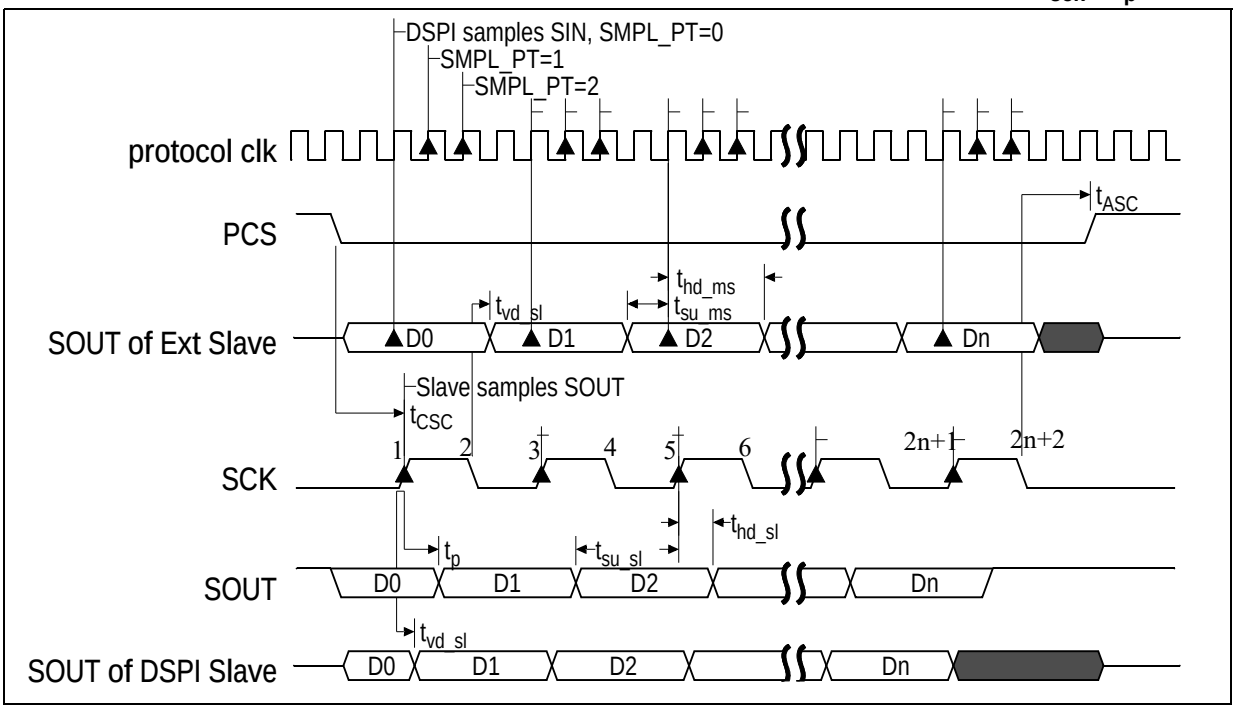


Figure 634. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/2$)

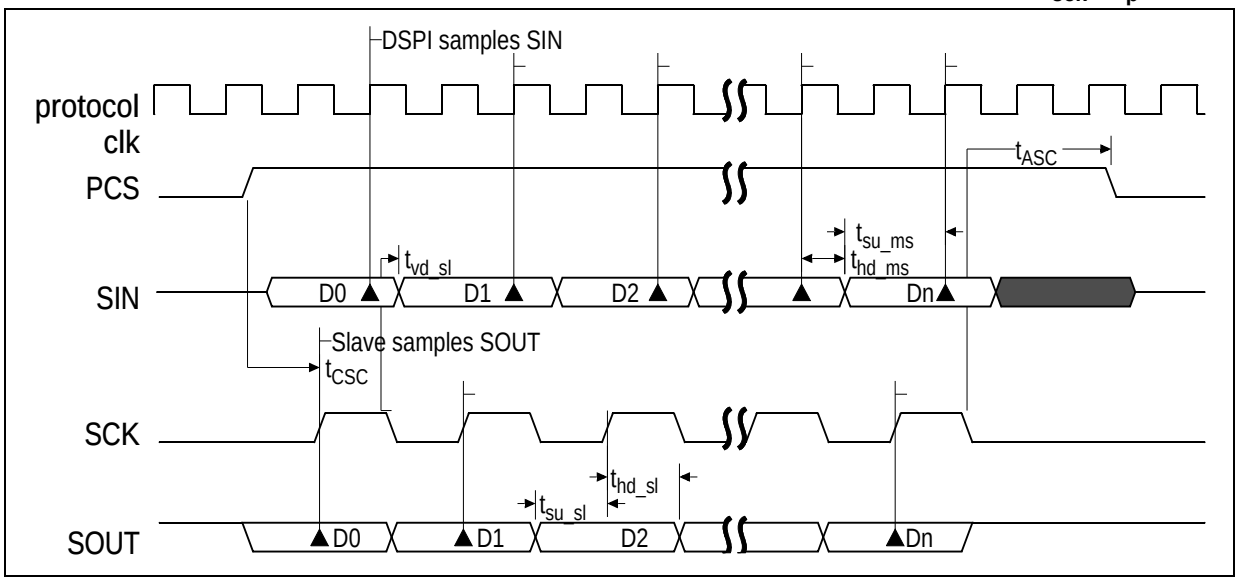
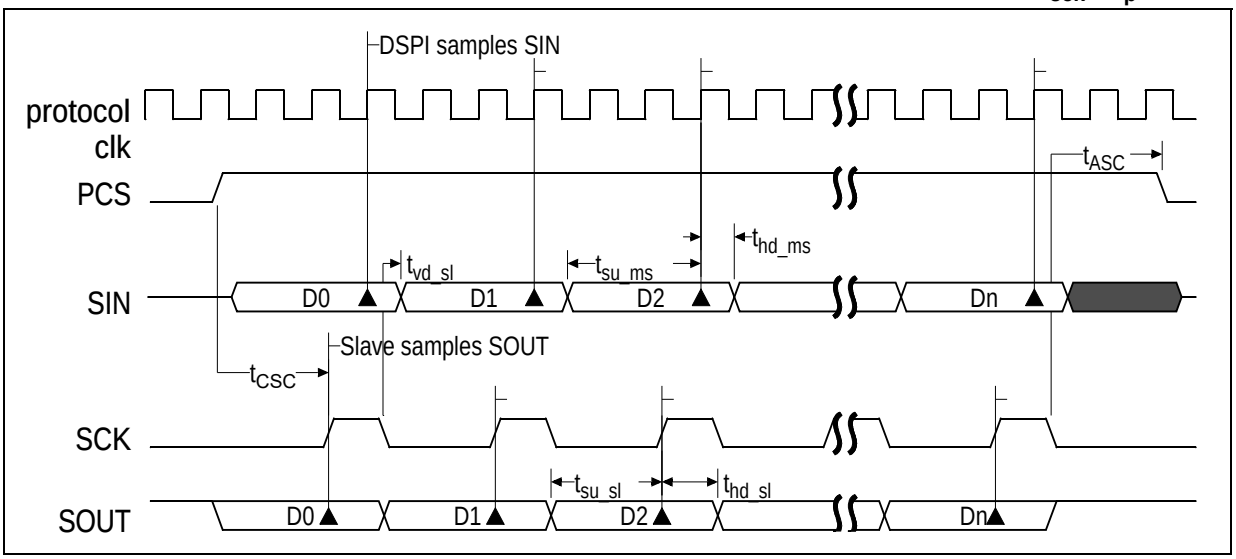


Figure 635. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/3$)



46.5.6.4 Modified SPI/DSI 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 636. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/2$)

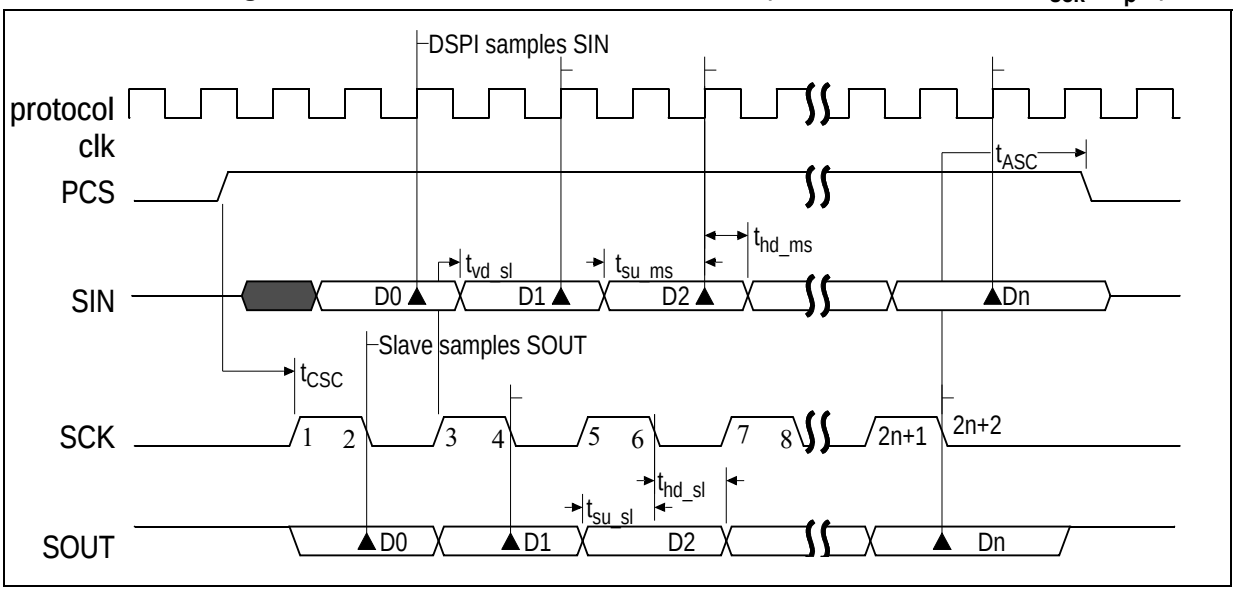


Figure 637. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/3$)

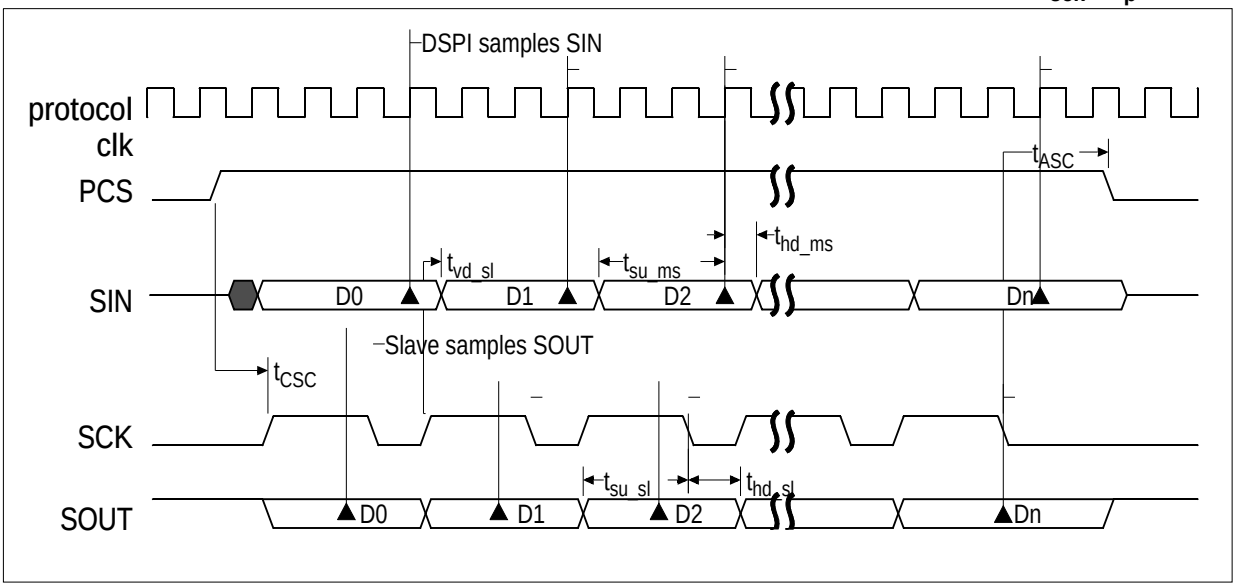
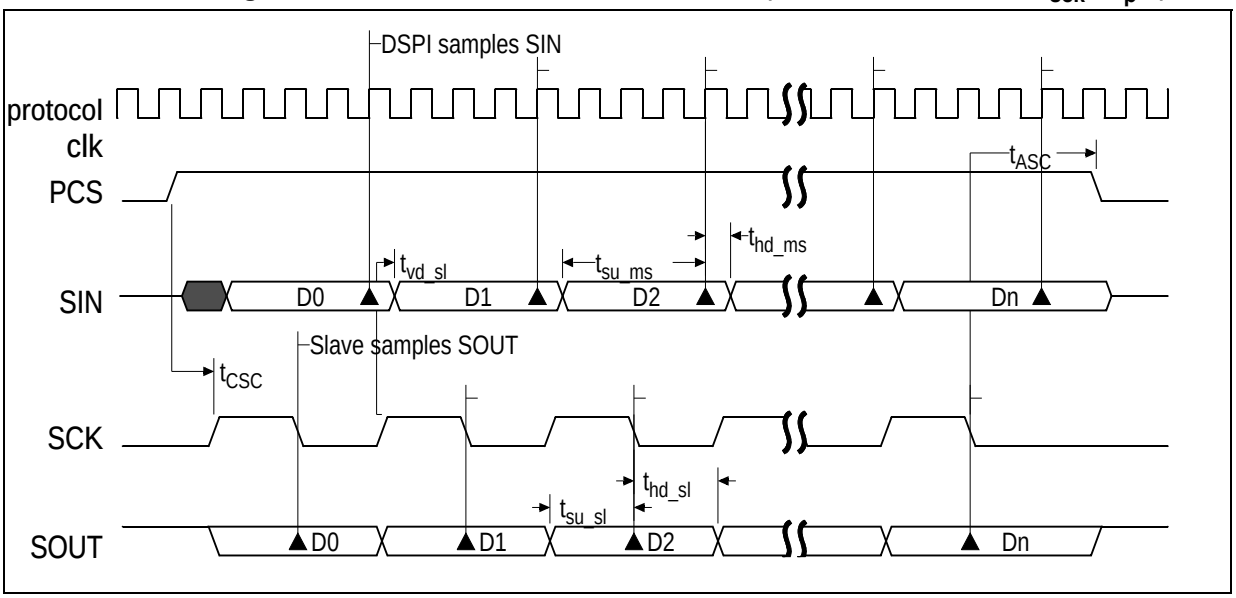


Figure 638. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/4$)



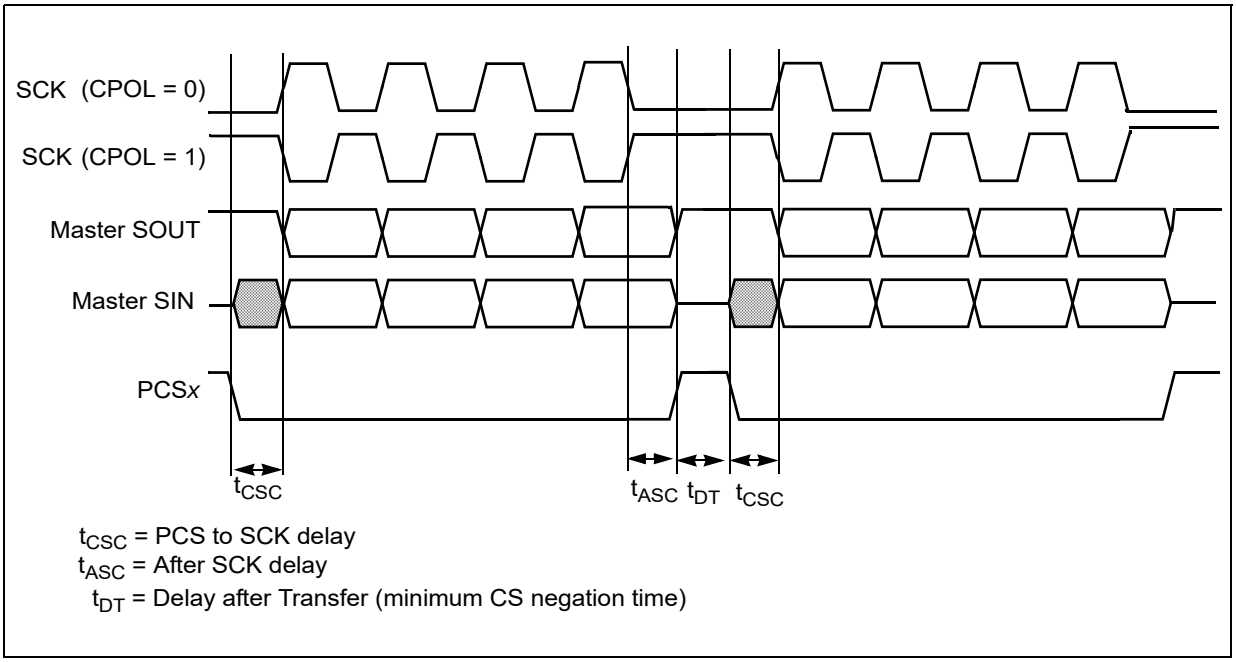
46.5.6.5 Continuous Selection Format

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. Continuous Selection is enabled for the DSI Configuration by setting the DCONT bit in the DSICR0.

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

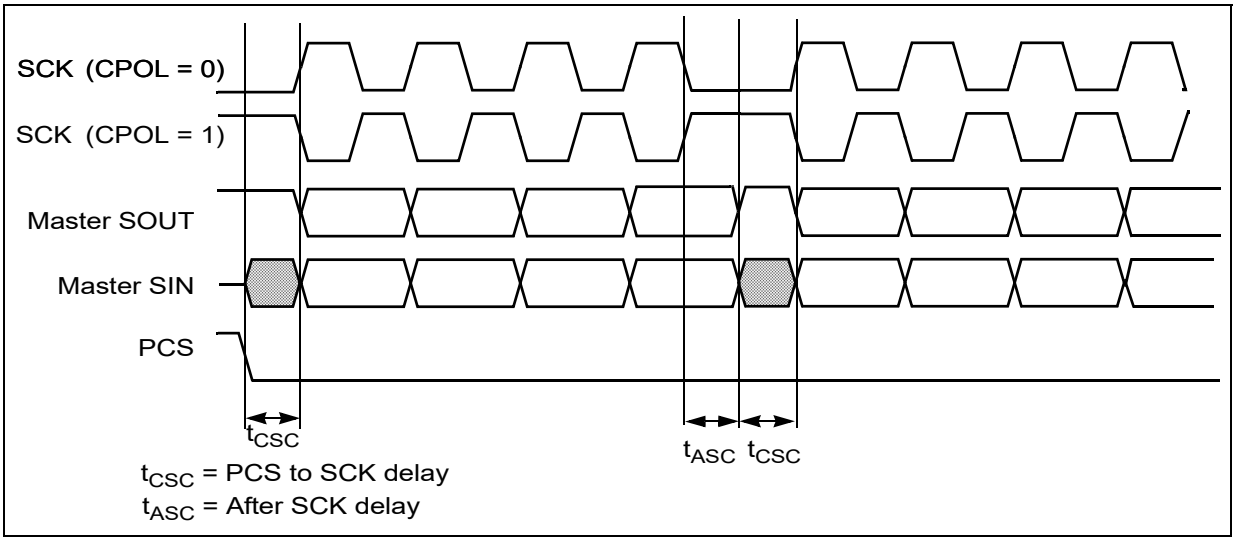
signals are selected by the PCSISn bits in the MCR. The timing diagram in *Figure 639* is for two four-bit transfers with CPHA = 1 and CONT = 0.

Figure 639. 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 640* shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

Figure 640. 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]/DSICR0[DCONT] bits must be de-asserted before asserting MCR[HALT] bit (in master mode). This ensures 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.

Note: 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.

46.5.6.6 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. Masking of delays is not allowed in DSI and CSI configurations or if the transfer is non-continuous.

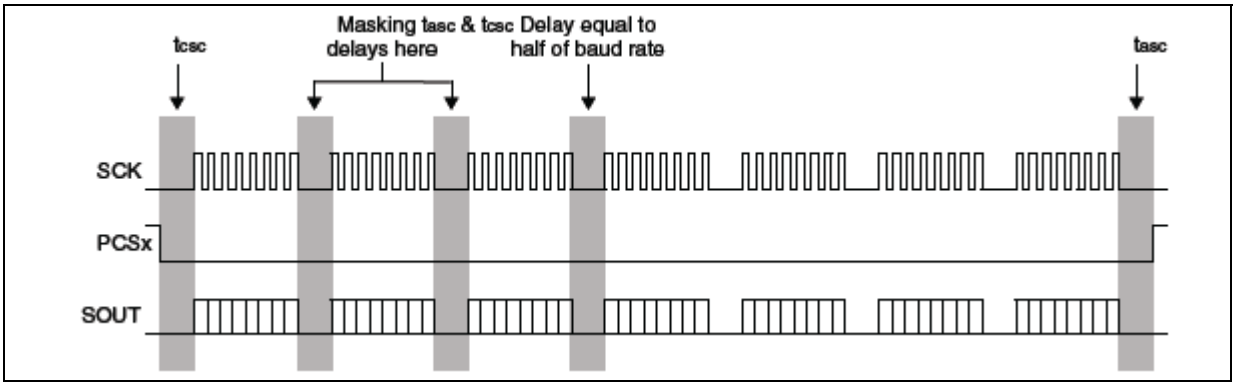
The Fast Continuous Selection Format is enabled by writing ‘1’ into FCPCS bit of the MCR. When this bit is asserted, MASC and MCSC bits of the PUSHR perform the function of mask bits for the transmit frame. These bits individually mask the t_{ASC} and t_{CSC} delays as programmed by the user software. 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 641 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 641. 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.

46.5.7 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 the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous SCK only if MCR[HALT] bit is low. 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.
- When the DSPI is in DSI configuration, the CTAR specified by the DSICTAS field is used at all times.
- When the DSPI is in CSI configuration, the CTAR selected by the DSICTAS field is used initially.
  - At the start of a SPI frame transfer, the CTAR specified by the CTAS value for the frame is used.
  - At the start of a DSI frame transfer, the CTAR specified by the DSICTAS field is used.
- In all configurations, 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.

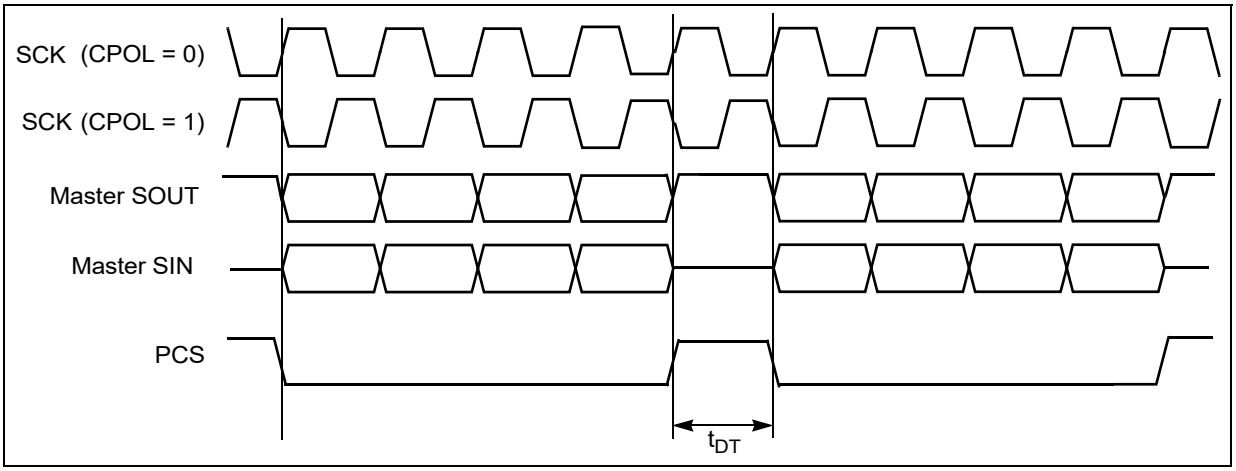
It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle.

When TSB configuration is enabled the t_{DT} is programmable from 1 to 64 SCK cycles. *Figure 642* 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 642. Continuous SCK timing diagram (CONT = 0)



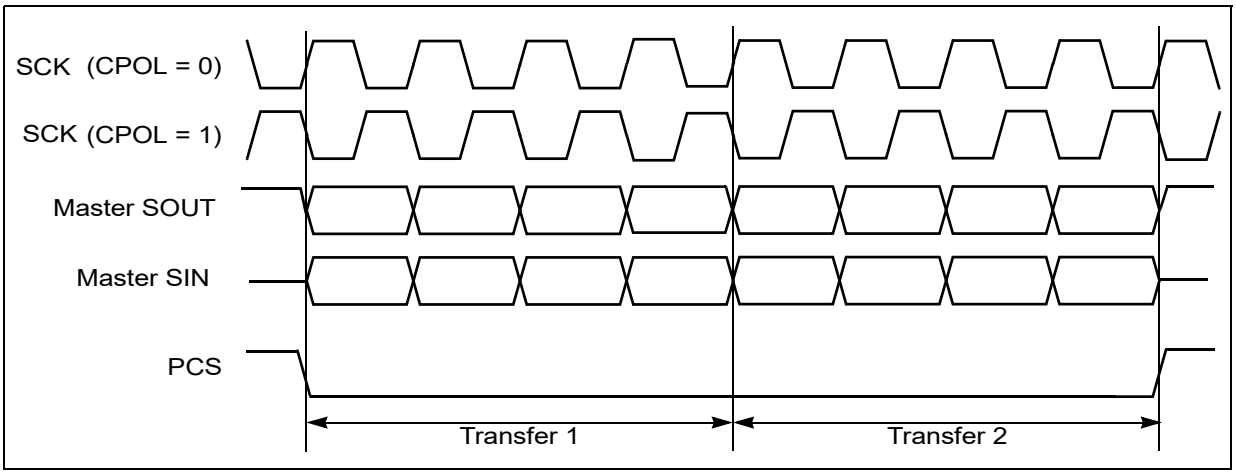
If the CONT bit in the TX FIFO entry is set or the DCONT in the DSICR0 is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with

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 46.5.1: Start and Stop of DSPI transfers*.
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

Figure 643 shows the timing diagram for Continuous SCK format with Continuous Selection enabled.

Figure 643. Continuous SCK Timing Diagram (CONT = 1)



46.5.8 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.

46.5.9 Timed Serial Bus (TSB)

The DSPI can be programmed in Timed Serial Bus configuration by setting DSICR0[TSBC] bit. See DSPI DSI Configuration Register 0 (DSPI_DSICR0) for details.

TSB configuration provides the Micro Second Channel (MSC) downstream channel support.

The MSC upstream channel is not supported by the DSPI, but can be supported by any available Serial Communication Controller (SCI or UART) in the device.

To work in TSB mode the DSPI must be in master mode and in DSI (DCONF = 0b01) or CSI (DCONF = 0b10) configuration. Both Continuous and Non Continuous Serial Communication Clock (controlled by the DSPI_MCR[CONT_SCKE] bit) are supported in the TSB mode.

Figure 644 shows the signals used in the TSB interface.

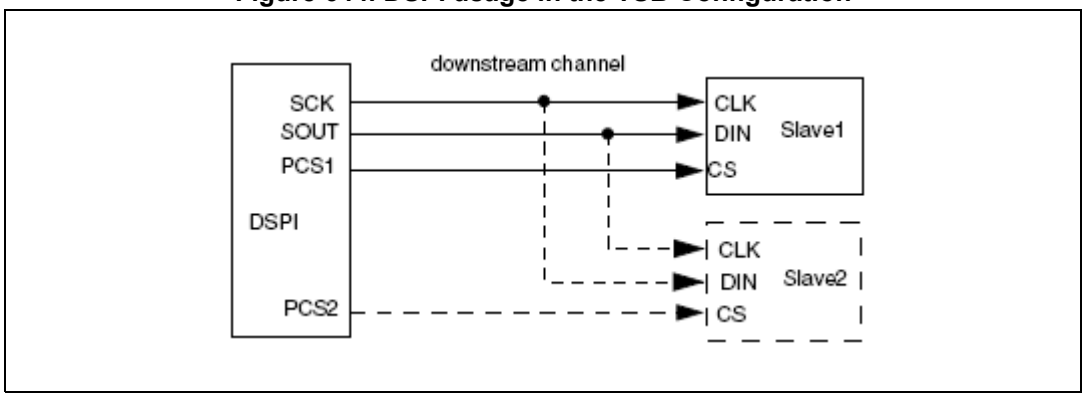
In the TSB configuration the DSPI is able to send from 4 to 66 bits MSC data frames (4 to 64 serialized data bits and up to 2 Data Selection zero bits). The serialized data bits source can be either:

- the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR0/1), written by the host software,
- parallel input pin states latched into the DSPI DSI Serialization Data Register (DSPI_SDR0/1).

DSICR0[TXSS] bit or DSPI_SSR0/1 register bits define the source of the data.

The Least Significant Bits of the DSPI_ASDR0 or DSPI_SDR0 registers are selected to be serialized if the data frame is set to less than 32 bits.

Figure 644. DSPI usage in the TSB Configuration



The PCS signals are driven together with SOUT. The t_{CSC} and t_{ASC} delays are not available. Delay after Transfer (DT) is set in SCK clock periods as a binary number formed by concatenation of the DSPI_CTARn PDT and DT fields plus one, allowing to set DT from 1 to 64 serial clock periods. DT field provides least significant bits and PDT field provides most significant bits of the Delay after Transfer.

Figure 645. TSB Downstream Frames

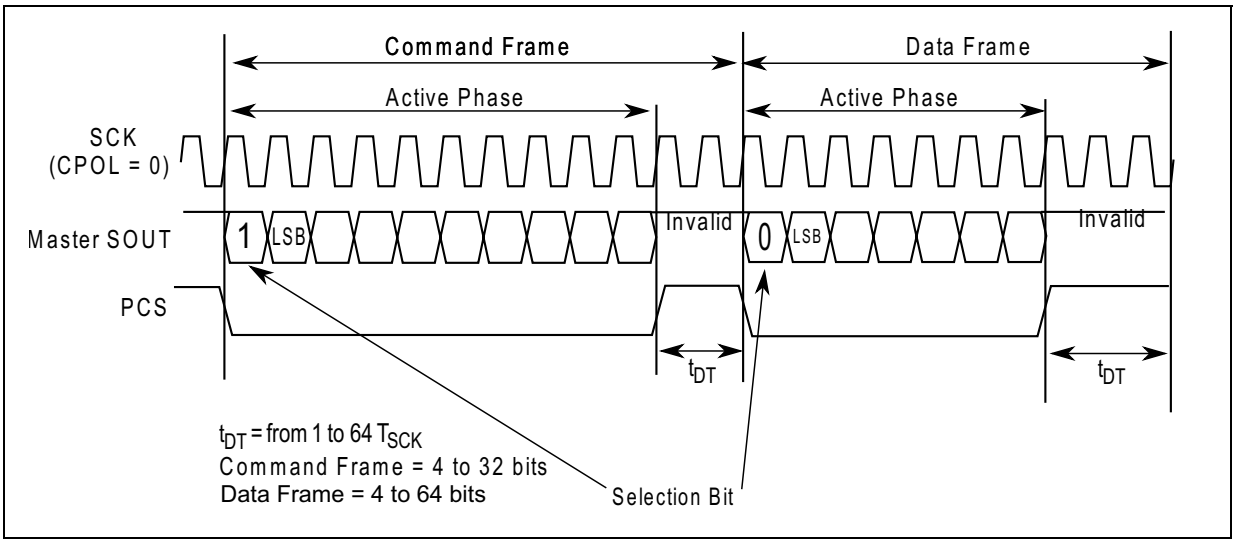


Figure 645 shows the two types of MSC downstream frames: command frame, and data frame.

The first transmitted bit, called the selection bit, determines the frame type:

- The selection bit ‘0’ indicates a data frame
- The selection bit ‘1’ indicates a command frame

Data frame may contain up to 2 selection bits to support two external slave devices, (so called dual receiver configuration) or no selection bits at all.

The command frame can be written by software, through SPI TX FIFO, using one or two FIFO entries with help of the CONT bit or MCR[XSPI]. The data frame consists of up to 64 bits from the SDR1/0 or ASDR1/0 registers and up to two zero selection bits. Number of data bits in the data frame is defined by the DSCICR1[TSBCNT] field.

The selection bit of the MSC command frames (1) can be implemented by software.

The selection bits in the data frames are enabled by DSPI_DSICR1 DSE0 and DSE1 bits. Each DSEn bit set increases the data frame size by one bit.

To comply with MSC specification, set DSPI_CTARn[LSBFE] to transmit least significant bit first.

Regardless of the LSBFE bit setting, the Data Frame Selection Bits, if enabled, are always transmitted first, before the corresponding data subframes.

46.5.9.1 MSC Dual Receiver Support with PCS Switchover

When in TSB mode it is possible to switch the set of PCS signals that are driven during the first part of the frame to a different set of PCS signals during the second part of the frame. The bit, at which this switchover occurs, is defined by FMSZ field of the DSPI_CTARn register, selected by DSICTAS field of the DSICR0 register. If DSICR1[DSI64E] is enabled, then the bit at which the PCS switch occurs, is defined by the concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]} fields.

Note: When using Dual receiver support, it must be ensured that the number of bits to be transmitted to each slave must be at least 4.

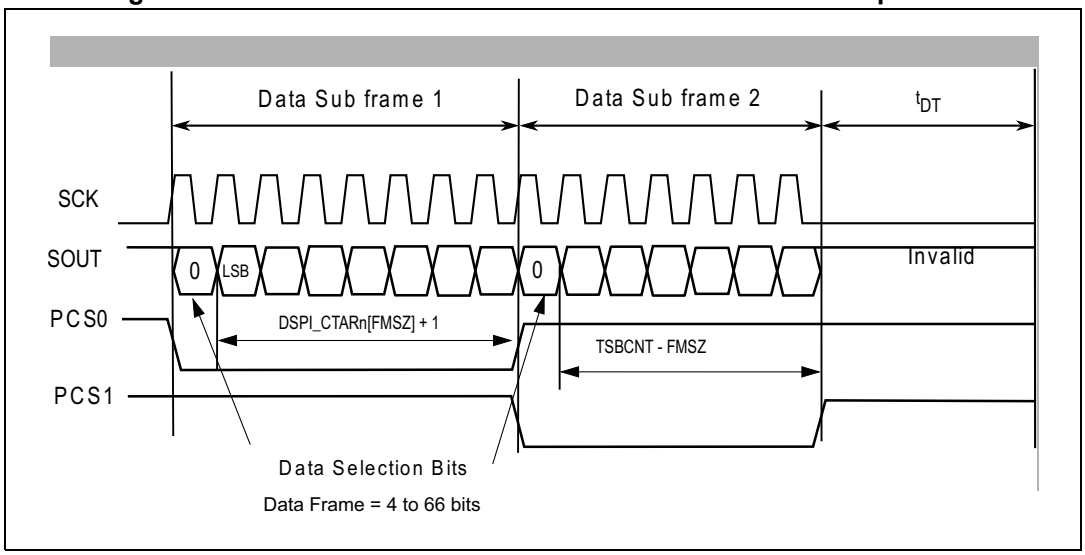
Number of the bits, not including the Data Selection Bit, in the first part of the frame is equal to value of the FMSZ field plus one. (This value changes to {FMSZ4,FMSZ} plus one, when DSICR1[DSI64E] is enabled). During this part of the frame the PCS signal levels are controlled by DSPI_DSICR0 DPCSn bits, after that by DSPI_DSICR1 DPCS1_n bits.

The PCS switchover occurs at the driving edge of the SCK clock output.

The second Data Selection Bit is inserted after the PCS switchover if enabled.

The Data Frame with PCS switchover is shown in *Figure 646*.

Figure 646. TSB Data Frame Format for MSC Dual Receiver Operation



46.5.10 Interleaved TSB (ITSB) Mode

This mode is similar to the TSB in all aspects except for the differences mentioned in *Table 669*. In the Interleaved TSB or ITSB mode, frame transmission is governed by the following rules:

- Transmission of frames is executed on a periodic trigger that is generated internally to the DSPI. The trigger period is calculated as follows:
  - Trigger Period = 64 bits (maximum data frame size) + 2 selection bits (maximum bits in data frame) + t_{PT}
    - t_{PT} is the passive frame time or dead time inserted to have constant trigger period.
  - The user software shall configure the trigger period and trigger source.
- If the previously transmitted frame was a SPI frame or the TX FIFO (or CMD FIFO) is empty, the frame transmitted on subsequent trigger will always be a DSI frame.
- If the previously transmitted frame was not a SPI frame and the TX_FIFO (and CMD FIFO) is not empty, the frame transmitted on subsequent trigger will always be a SPI frame.

This mode does not change the way DSI and SPI frame transmissions are done except that ITSB mode interleaves the source (SPI or DSI) in such a way to prevent back to back SPI frames. All features of SPI and DSI mode can be used except for the Continuous Frame Selection format (PUSHR[CONT] and DSICR0[DCONT] bits cannot be asserted).

MCR[MTFE] should be set to 0 during ITSB Mode of operation, since MicroSecond Channel upstream operation is not supported.

Table 669 lists the differences between the TSB and ITSB modes.

Table 669. Differences between TSB and ITSB modes

Mode attribute	TSB mode	ITSB mode
Frame priority	Priority available to frames from SPI. Back to back SPI frames are sent out.	No priority. Messages are interleaved such that there are no back-to-back SPI frames
Frame selection bit	Selection bit inserted for DSI frames only. Selection bit for SPI frames is inserted by software.	Selection bit for both DSI and SPI frames inserted automatically. Encoding of selection bit same as TSB mode (0 = DSI; 1 = SPI).
Frame transmission	Occasionally triggered transmission (that is trigger is received via hardware trigger input).	Periodically triggered transmission (that is frames are transmitted continuously on every trigger).
Frame gap	Gap between frames is configurable. Time between frames is t_{DT} which is configurable from 1 to 64 baud rate clock cycles.	Gap between frames is inserted automatically as the whole operation is based on a periodic trigger. If a frame has less than 64 bits to be transmitted, dead or passive time is automatically inserted by waiting for the next trigger pulse. Time t_{DT} is not used in this mode as it can be made part of the trigger period (as passive time).

In TSB and ITSB mode, separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI. MSC dual receiver support with PCS switchover is also supported in ITSB mode. See *Section 46.5.9.1: MSC Dual Receiver Support with PCS Switchover* for more details.

The ITSB mode is suitable to implement the Downstream Channel for the MSC Bus because of the ITSB mode features listed in the table above.

46.5.10.1 Configuring DSPI for ITSB mode

The following steps give the recommended way to initialize the DSPI for the ITSB mode of operation:

- The DSPI should be put in HALT mode (by asserting the MCR[HALT] bit) before programming the registers.
- Set the DSICR0[TSBC] and DSICR0[ITSB] bits. The DSICR0[CID] bit should be cleared. DSICR0[DCONT] should not be used in ITSB mode. If DSICR0[ITSB] bit is set without setting DSICR0[TSBC], then the DSPI operates in the Normal Mode depending on MCR[DCONF] bit.
- Enable frame completion interrupts for either any frame completion (TCF) or separate frame completion (DSITCF and SPITCF) as required by application.
- When remainder configuration of DSPI is complete, clear the MCR[HALT] bit to allow DSPI to start operations.

Once configured and DSPI is brought out of HALT, the ITSB mode shall start sequencing the frames from either DSI or SPI as per the rules mentioned in *Section 46.5.10: Interleaved TSB (ITSB) Mode*.

46.5.10.2 Using ITSB mode for MSC downstream channel

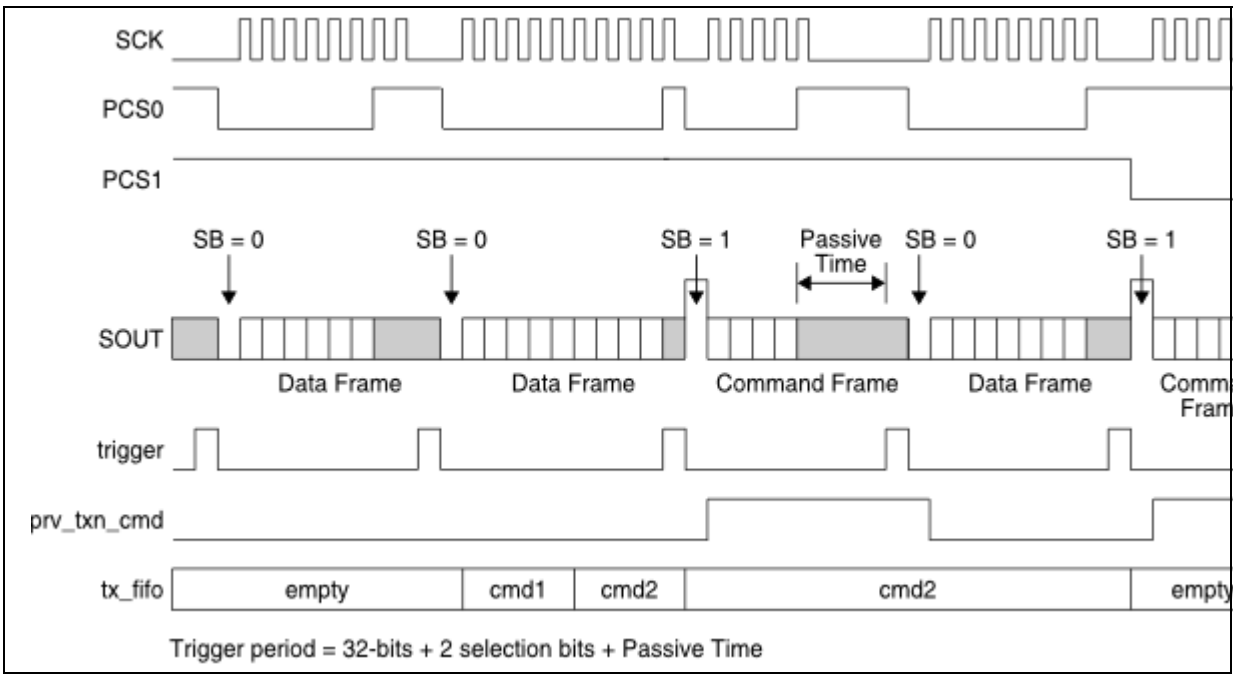
MSC Downstream Channel uses two types of frames: command frames and data frames. The Downstream operation of these frames require:

- Data frames can be sent back to back.
- Back to back command frames are not a legal scenario.
- If command frame has less bits than data frame, a gap is inserted to ensure next data frame starts at known reference time.
- In dual MSC ASIC, some bits of the data frame (which can be up to 64 bits) goes to first ASIC and remainder bits to second ASIC.
- In dual MSC ASIC, if a command frame is sent to one ASIC, it must not change the timing of the next data frame.

The ITSB mode can help implement the MSC Downstream Channel by allowing the command frames to be sent via the SPI interface and data frames to be sent via the DSI interface. Once the DSPI has been configured to work in ITSB mode (See *Section 46.5.10.1: Configuring DSPI for ITSB mode*), the user software can program the DSI and SPI sides to send data and command frames respectively. The programming of frames for transmission via DSI or SPI is same as explained in *Section 46.5.4: Combined Serial Interface (CSI) configuration*.

Figure 647 shows a sample MSC Downstream transmission using the ITSB mode.

Figure 647. Sample MSC downstream transmission using ITSB mode



Brief description of operation includes:

- At the first trigger, since there are no command frames, a data frame is transmitted: to send a command frame at the start of operation, the user software must program a command frame in the TX_FIFO before enabling DSPI (MCR[HALT] = 0).
- The second trigger sets off the transmission of another data frame.
- Between second and third trigger, two command frames are pushed into the TX_FIFO.
- At third trigger, a command frame is transmitted as the TX_FIFO is not empty: since only 16 bits or less are transmitted, no transmission happens till next trigger. This is marked as the “Passive Time”. The passive time is added in all frames after frame transmission completes and until next trigger pulse is detected.
- At fourth trigger, since the previous transmission was a command frame, a data frame is transmitted.
- At fifth trigger, since one command frame is pending in TX_FIFO and last transmitted frame was not a command frame, a command frame is transmitted.
- At sixth trigger (not shown), the process continues.

46.5.11 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 or/and DSI frame transmission in case of a frame reception with parity error.

46.5.11.1 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 sub-frame 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 sub-frame 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.

46.5.11.2 Parity for DSI frames

When DSPI is in Master Mode, parity generation is controlled by PE and PP bits of the DSPI_DSICR0 register similar to the SPI frames. The parity is calculated and checked for each DSI frame. (DSPI_DSICR0[DCONT] bit has no effect on parity generation. In slave mode, the parity is controlled by the PE and PP bits of the CTAR1 register.

If the parity error occurs for received DSI frame, the DSPI_SR[DPEF] bit is set. If DSPI_DSICR0[PES] bit is set, the DSPI stops DSI frames transmission. To resume DSI operation, clear the DSPI_SR[DPEF] or the DSPI_DSICR0[PES] bits.

46.5.12 Interrupts/DMA requests

Certain DSPI conditions can generate only interrupt requests and other DSPI conditions can generate either interrupt requests or DMA requests. *Table 670* lists these conditions.

Table 670. 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	—
DSI Transfer Complete	DSITCF	Yes	—
TX FIFO Underflow	TFUF	Yes	—
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	—
SPI Parity Error	SPEF	Yes	—
DSI Parity Error	DPEF	Yes	—
DSI Deserialized Data Match	DDIF	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.

46.5.12.1 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 the EOQF_RE bit in the RSER 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.

46.5.12.2 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.

Note: 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.*

46.5.12.3 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.

Note: 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.*

46.5.12.4 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 TFIWF_RE bit is set in the RSER.

46.5.12.5 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 the TCF_RE bit is set in the RSER.

46.5.12.6 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 the CMDTCF_RE bit is set in the RSER.

46.5.12.7 SPI Transfer Complete interrupt request

The SPI Transfer Complete Request indicates the end of transfer of an SPI frame in any Transmission Mode which uses DSPI in CSI Mode. The SPI Transfer Complete Request is generated when SR[SPITCF] flag asserts and SPITCF_RE bit is set in the RSER.

46.5.12.8 DSI Transfer Complete interrupt request

The DSI Transfer Complete Request indicates the end of transfer of a DSI frame in any Transmission Mode which uses DSPI in CSI Mode. The DSI Transfer Complete Request is generated when SR[DSITCF] flag asserts and DSITCF_RE bit is set in the RSER.

46.5.12.9 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.

46.5.12.10 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 RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

46.5.12.11 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 RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

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.

46.5.12.12 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.

46.5.12.13 DSI Frame Parity Error Interrupt Request

The DSI Frame Parity Error Flag indicates that a DSI frame with parity error has been received. The DPEF_RE bit in the DSPI_RSER must be set for the interrupt request to be generated.

46.5.12.14 Deserialized Data Match Interrupt Request

The Deserialized Data Match Flag (DDIF) indicates that a DSI frame matches DSPI_DPIR0/1 data, masked with DSPI_DIMR0/1, has been received. This Request is generated if DDIF_RE bit in the DSPI_RSER is set.

46.5.13 Power saving features

The DSPI supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory-mapped logic

Note: When in Slave Mode, the module can only be put in Low Power Modes (such as STOP and HALT) once the Master module has transitioned to the Low Power Mode as well.

46.5.13.1 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.

If a DSPI block is configured in Slave mode, External Stop mode requests must be run after the last clock cycle or Low-power mode is not entered.

46.5.13.2 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 the MDIS bit in the MCR. The module disable mode can also be initiated by hardware.

When the MDIS bit 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.
- 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.
- Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect.
- 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.

46.6 Initialization/application information

This section describes how to initialize the DSPI module.

46.6.1 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 the CLR_TXF bit in the MCR. Flush RX FIFO by writing a ‘1’ to the CLR_RXF bit in the MCR.
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 SPI_TCNT field in the TCR.
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.

46.6.2 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].

46.6.3 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.

46.6.4 Baud rate settings

Table 671 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 671. Baud rate values (bps)

Baud Rate scaler values (DSPI*n*_CTAR[BR])	Baud rate divider prescaler values (DSPI*n*_CTAR[PBR])			
	2	**3**	**5**	**7**
2	25.0 MHz	16.7 MHz	10.0 MHz	7.14 MHz
4	12.5 MHz	8.33 MHz	5.00 MHz	3.57 MHz
6	8.33 MHz	5.56 MHz	3.33 MHz	2.38 MHz
8	6.25 MHz	4.17 MHz	2.50 MHz	1.79 MHz
16	3.12 MHz	2.08 MHz	1.25 MHz	893 kHz
32	1.56 MHz	1.04 MHz	625 kHz	446 kHz
64	781 kHz	521 kHz	312 kHz	223 kHz
128	391 kHz	260 kHz	156 kHz	112 kHz
256	195 kHz	130 kHz	78.1 kHz	55.8 kHz
512	97.7 kHz	65.1 kHz	39.1 kHz	27.9 kHz
1024	48.8 kHz	32.6 kHz	19.5 kHz	14.0 kHz
2048	24.4 kHz	16.3 kHz	9.77 kHz	6.98 kHz
4096	12.2 kHz	8.14 kHz	4.88 kHz	3.49 kHz
8192	6.10 kHz	4.07 kHz	2.44 kHz	1.74 kHz
16384	3.05 kHz	2.04 kHz	1.22 kHz	872 Hz
32768	1.53 kHz	1.02 kHz	610 Hz	436 Hz

46.6.5 Delay settings

Table 672 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.

This table does not apply for TSB Continuous mode.

Table 672. 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

46.6.6 Calculation of FIFO pointer addresses

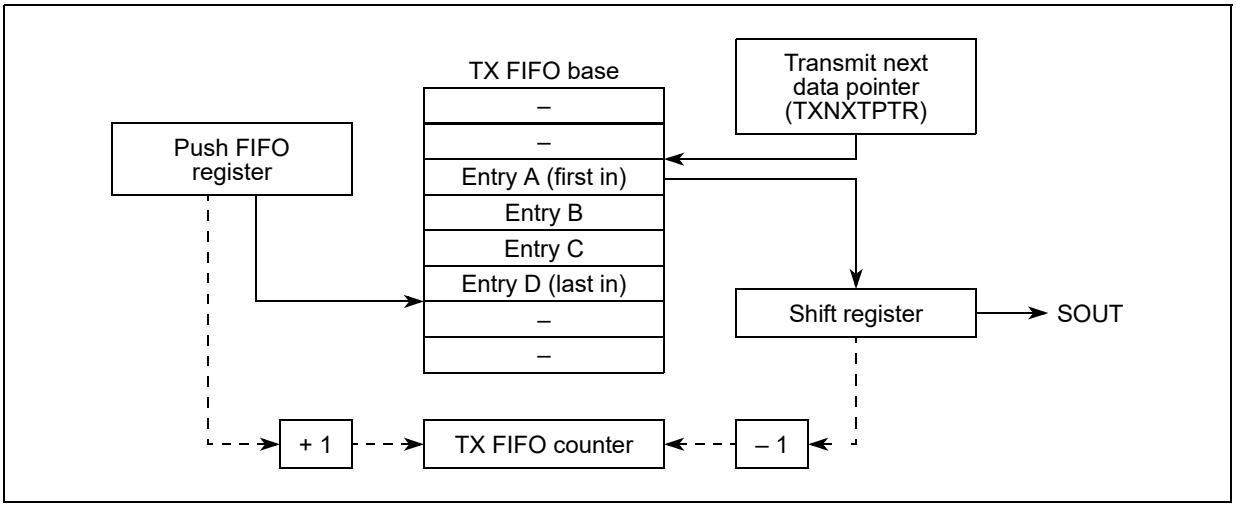
Complete visibility of the TX and CMD and RX FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and a memory-mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory-mapped.

- For TX FIFO, the first-in pointer is the Transmit Next Pointer (TXNXTPTR).
- For CMD FIFO, the first-in pointer is the Command Next Pointer (CMDNXTPTR).
- For RX FIFO, the first-in pointer is the Pop Next Pointer (POPNXTPTR).

Figure 648 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 are applicable to the CMD FIFO and RX FIFO.

See *Section 46.5.2.4: Transmit First In First Out (TX FIFO) buffering mechanism*, *Section 46.5.2.5: Command First In First Out (CMD FIFO) buffering mechanism* and *Section 46.5.2.6: Receive First In First Out (RX FIFO) Buffering Mechanism* for details on the FIFO operation.

Figure 648. TX FIFO pointers and counter



46.6.6.1 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 25
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 26
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

46.6.6.2 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 27
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 28
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

46.6.6.3 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 29
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 30
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

47 LVDS Fast Asynchronous Serial Transmission – Interprocessor Communications

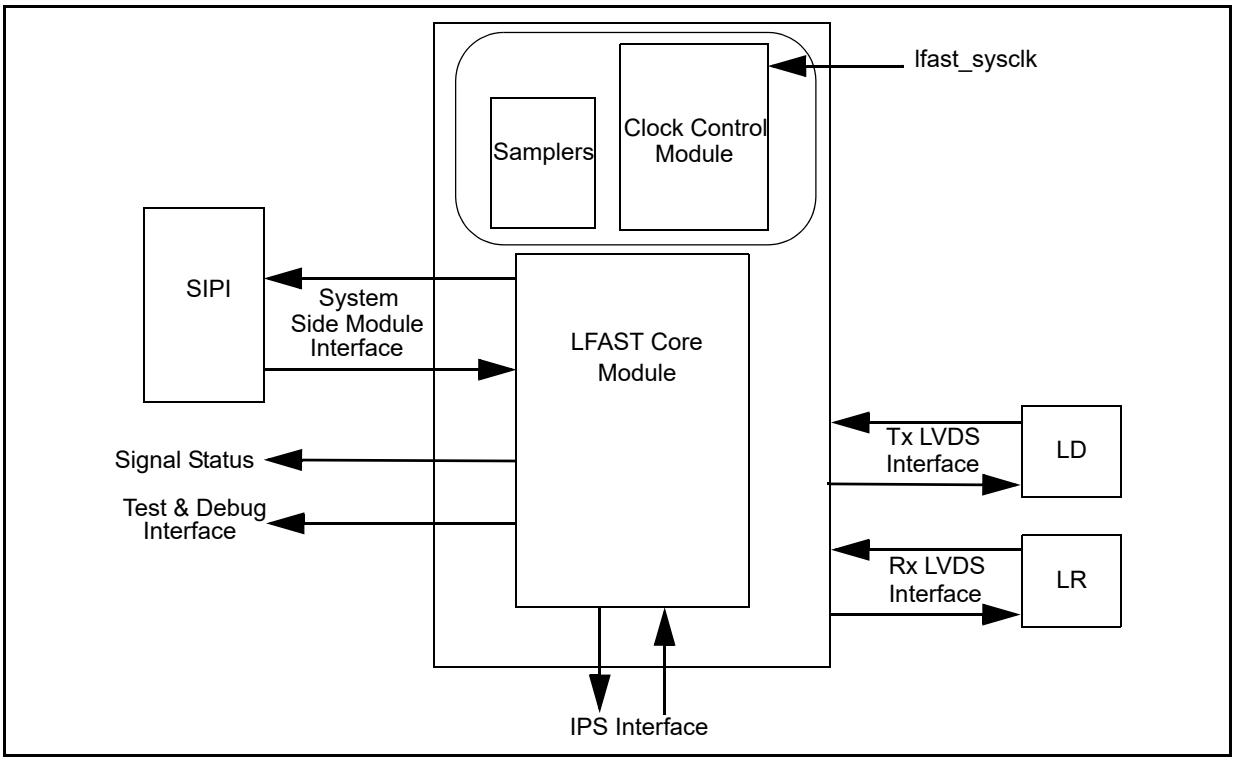
47.1 Introduction

This chapter describes the specifications of the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in dual mode (software configurable master/slave operation) for interprocessor communications.

47.2 Block diagram

Figure 649 depicts LFAST interaction with other modules on the device.

Figure 649. LFAST block diagram



47.3 External signals

LFAST is a five pin interface with the following signals:

- lfast_sysclk
  - Reference clock of the LFAST master and slave
- txdatap/txdatan
  - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
  - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface.

47.3.1 LFAST operating data rates

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps (lfast_sysclk ÷ 4 or lfast_sysclk ÷ 2) is referred to as low data rate, and the high data rate is in the Data Sheet.

47.4 LFAST frame structure

LFAST frames are transmitted and received across a serial interface using a packeted asynchronous communication format.

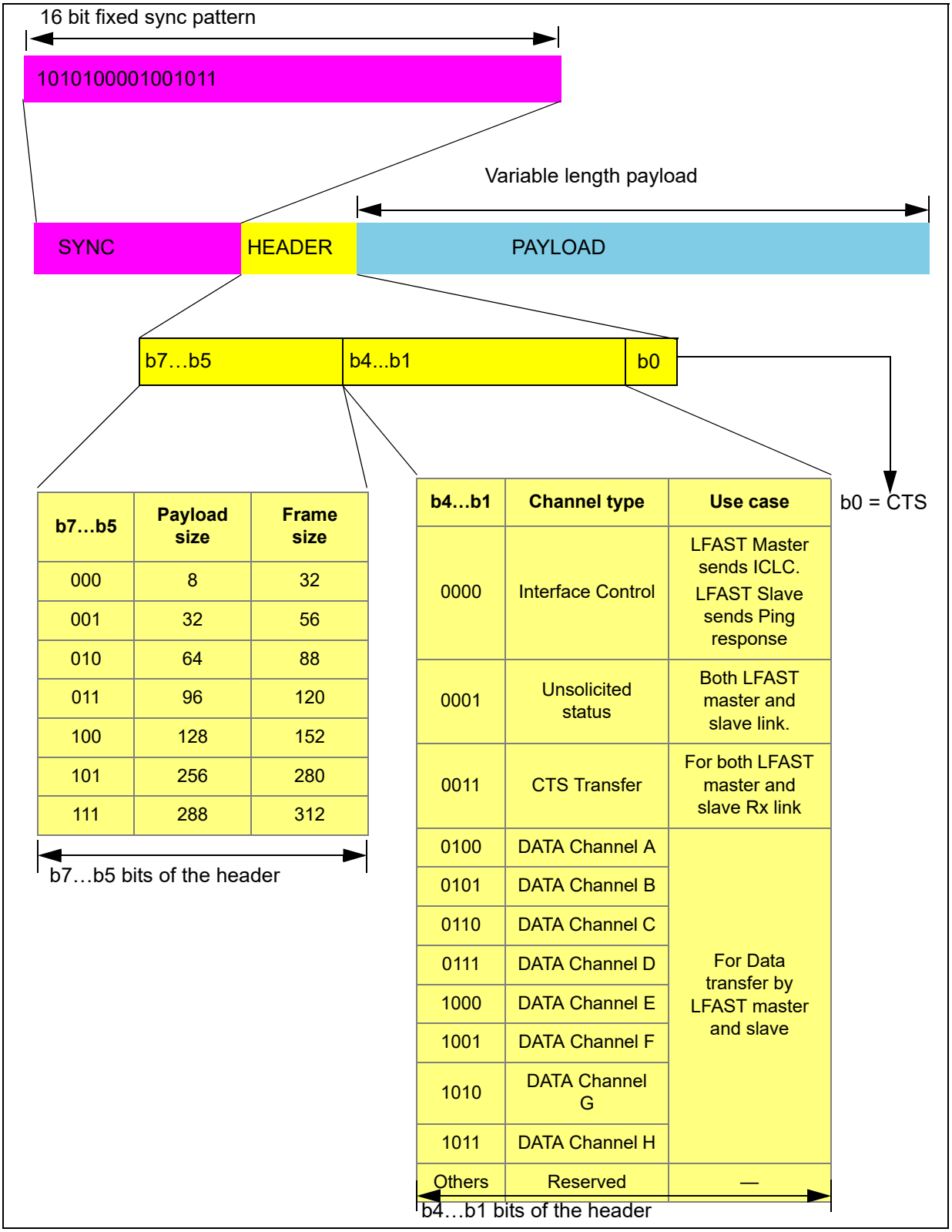
A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length. A sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See *Figure 650* for details of the LFAST frame.

Figure 650. LFAST frame structure



b7…b5	Payload size	Frame size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
111	288	312

b4…b1	Channel type	Use case
0000	Interface Control	LFAST Master sends ICLC. LFAST Slave sends Ping response
0001	Unsolicited status	Both LFAST master and slave link.
0011	CTS Transfer	For both LFAST master and slave Rx link
0100	DATA Channel A	For Data transfer by LFAST master and slave
0101	DATA Channel B	
0110	DATA Channel C	
0111	DATA Channel D	
1000	DATA Channel E	
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	—

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

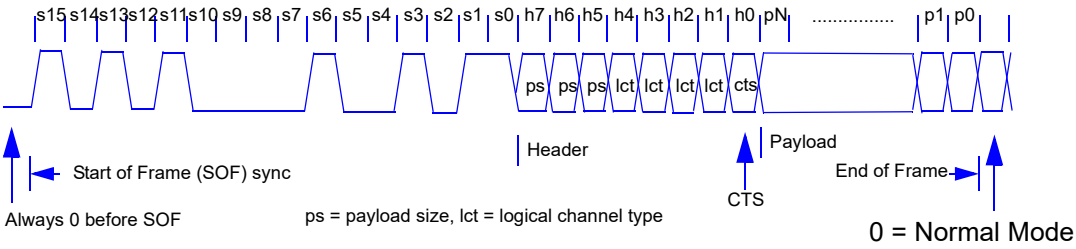


Figure 651. Serial frame structure

The frame structure is shown in *Figure 651* and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto-correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size according to the *Table 673*.

Table 673. Header payload sizes

b7-b5 (bin)	Payload Size	Frame Size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	—	—
111	288	312

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type and clear to send (CTS) frame.
- **Header - (b0):** CTS on the both LFAST master and slave devices.
- **Payload: Content dependent upon frame type.**
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode).

47.5 Features

- Supports dual mode (register configurable Master/Slave).
- Supports asynchronous data transfer rates up to the maximum data rated shown in the product Data Sheet.
- Transmits and receives data, CTS, ICLC and unsolicited frames.
- Provision of five interrupts for Tx and Rx channels.
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes.
- Supports LFAST defined CTS controlled data transfer. No dependency between data transmission and reception unless CTS mode is enabled.
- Supports flow control using sliding window protocol.
- Provides configurable frame length for data frame with variable payload sizes of 32, 64, 96, 128, 256 or 288 bits.
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits.
- Supports PLL configuration (for example, feedback loop divider, etc.) through registers.
- Supports LVDS configuration through registers.
- Supports multiple loopback modes for checking the physical interface.
- Supports automatic ping response generation in slave mode.
- Supports for detection of unsupported channel number and unsupported payload size.

47.6 Memory map and register definition

All registers are 32 bits wide.

47.6.1 Memory map

Table 674. LFAST memory map

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
0000	LFAST Mode Configuration Register (MCR)	R/W	0000_0010	*on page 1191*
0004	LFAST Speed Control Register (SCR)	R/W	0001_0000	*on page 1192*
0008	LFAST Correlator Control Register (COCR)	R/W	0000_000E	*on page 1194*
000C	LFAST Test Mode Control Register (TMCR)	R/W	0000_0000	*on page 1195*
0010	LFAST Auto Loopback Control Register (ALCR)	R/W	0000_0000	*on page 1197*
0014	LFAST Rate Change Delay Control Register (RCDCR)	R/W	000F_0000	*on page 1198*
0018	LFAST Wakeup Delay Control Register (SLCR)	R/W	1201_5F02	*on page 1198*
001C	LFAST ICLC Control Register (ICR)	R/W	0000 0000	*on page 1200*
0020	LFAST Ping Control Register (PICR)	R/W	0000_80CA	*on page 1201*
0024–0028	Reserved	—	—	—
002C	LFAST Rx FIFO CTS Control Register (RFCR)	R/W	000F 0009	*on page 1202*

Table 674. LFAST memory map(Continued)

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
0030	LFAST Tx Interrupt Enable Register (TIER)	R/W	0000 0000	on page 1202
0034	LFAST Rx Interrupt Enable Register (RIER)	R/W	0000 0000	on page 1203
0038	LFAST Rx ICLC Interrupt Enable Register (RIIER)	R/W	0000 0000	on page 1205
003C	Reserved	—	—	—
0040	LFAST LVDS Control Register (LCR)	R/W	0000_5024	on page 1206
0044	LFAST Unsolicited Tx Control Register (UNSTCR)	R/W	0000_0000	on page 1208
0048	LFAST Unsolicited Tx Data Registers 8 (UNSTDR8)	R/W	0000_0000	on page 1209
004C	LFAST Unsolicited Tx Data Registers 7 (UNSTDR7)	R/W	0000_0000	on page 1209
0050	LFAST Unsolicited Tx Data Registers 6 (UNSTDR6)	R/W	0000_0000	on page 1209
0054	LFAST Unsolicited Tx Data Registers 5 (UNSTDR5)	R/W	0000_0000	on page 1209
0058	LFAST Unsolicited Tx Data Registers 4 (UNSTDR4)	R/W	0000_0000	on page 1209
005C	LFAST Unsolicited Tx Data Registers 3 (UNSTDR3)	R/W	0000_0000	on page 1209
0060	LFAST Unsolicited Tx Data Registers 2 (UNSTDR2)	R/W	0000_0000	on page 1209
0064	LFAST Unsolicited Tx Data Registers 1 (UNSTDR1)	R/W	0000_0000	on page 1209
0068	LFAST Unsolicited Tx Data Registers 0 (UNSTDR0)	R/W	0000 0000	on page 1209
006C–007C	Reserved	—	—	—
0080	LFAST Global Status Register (GSR)	R	0000_0000	on page 1210
0084	LFAST Ping Status Register (PISR)	R	0000_0000	on page 1211
0088–0090	Reserved	—	—	—
0094	LFAST Data Frame Status Register (DFSR)	R	0000_0000	on page 1211
0098	LFAST Tx Interrupt Status Register (TISR)	R/W	0000_0000	on page 1212
009C	LFAST Rx Interrupt Status Register (RISR)	R/W	0000_0000	on page 1213
00A0	LFAST Rx ICLC Interrupt Status Register (RIISR)	R/W	0000_0000	on page 1215
00A4	LFAST PLL and LVDS Status Register (PLLLSR)	R	0002_0003	on page 1216
00A8	LFAST Unsolicited Rx Status Register (UNSRSR)	R/W	0000_0000	on page 1217
00AC	LFAST Unsolicited Rx Data Register 8 (UNSRDR8)	R	0000_0000	on page 1218
00B0	LFAST Unsolicited Rx Data Register 7 (UNSRDR7)	R	0000_0000	on page 1218
00B4	LFAST Unsolicited Rx Data Register 6 (UNSRDR6)	R	0000_0000	on page 1218
00B8	LFAST Unsolicited Rx Data Register 5 (UNSRDR5)	R	0000_0000	on page 1218
00BC	LFAST Unsolicited Rx Data Register 4 (UNSRDR4)	R	0000_0000	on page 1218
00C0	LFAST Unsolicited Rx Data Register 3 (UNSRDR3)	R	0000_0000	on page 1218
00C4	LFAST Unsolicited Rx Data Register 2 (UNSRDR2)	R	0000_0000	on page 1218

Table 674. LFAST memory map(Continued)

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
00C8	LFAST Unsolicited Rx Data Register 1 (UNSRDR1)	R	0000_0000	*on page 1218*
00CC	LFAST Unsolicited Rx Data Register 0 (UNSRDR0)	R	0000_0000	*on page 1218*

The conventions in *Figure 652* and *Table 675* serve as a key for the register summary and individual register diagrams.

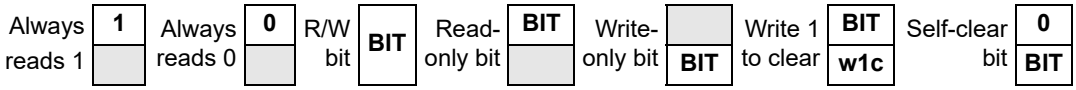


Figure 652. Key to Register Fields

Table 675 provides a key for register figures and tables and the register summary.

Table 675. 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 bits value (other than a hardware reset).
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but its cleared by the H/W once the operation defined by the bit is over
Reset Values	
0	Resets to zero.
1	Resets to one.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

47.6.2 Register descriptions

47.6.2.1 Mode Configuration Register (MCR)

Offset: 0000h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSEN(2)	0	0	0	0	0	0	IPGDBG	0	0	0	0	0	0	0	LSSEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DRFEN	RXEN	TXEN	0	0	0	0	0	0	0	0	TXARBD	CTSEN(3)	0	DRFRST(1)	DATAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1 Set by user software and cleared by system hardware.

2 Writable only once after the asynchronous reset.

3 Only writable when MCR[DRFEN] = 0.

Figure 653. Mode Configuration Register (MCR)

Table 676. MCR field descriptions

Field	Description
0 MSEN	LFAST Master or Slave mode Enable. This bit selects either the LFAST master or slave functionality. This bit can be written only once after the asynchronous reset. 0 Enable the modules LFAST Slave functionality only. 1Enable the modules LFAST Master functionality only.
1-6	Reserved
7 IPGDBG	Control bit to enable support for IPG Debug mode. This mode is indicated by assertion of IPG debug mode signal. 0 IPG debug mode enable signal will be ignored. 1 IPG debug mode enable signal will not be ignored.
8-14	Reserved
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode. Refer to *Section 47.10.2: Slow speed clock* for more detailed description. 0 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable. This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.

Table 676. MCR field descriptions(Continued)

Field	Description
17 RXEN	LFAST Receiver Enable. This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS LR. 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface is not disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable. This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be set and cleared by LFAST slave H/W on reception of an ICLC command frame. 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 Transmitter is Enabled.
19-26	Reserved
27 TXARBD	Tx Arbiter Disable. This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
28 CTSEN	CTS Enable. This bit defines the Push-Pull mode of the LFAST devices receiver. This bit is used to enable/disable CTS mode of the Tx block. 0 CTS mode is disabled. Indicates that the device is in Push mode. The CTS bit of frames transmitted is 1. The CTS bit does not represent the status of Rx FIFO. 1 CTS mode is enabled. The CTS bit of all transmit frames is set when the Rx FIFO empty space is on or above higher threshold, and cleared when the Rx FIFO empty space is on or below lower threshold.
29	Reserved
30 DRFRST	LFAST Soft Reset. This bit is automatically cleared after reset. 0 No soft reset. 1 Soft reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable. This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

47.6.2.2 Speed Control Register (SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Offset: 0004h 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	DRMD(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	RDR	0	0	0	0	0	0	0	TDR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 Only writable when MCR[DRFEN] = 0.

Figure 654. Speed Control Register (SCR)

Table 677. SCR field descriptions

Field	Description
0-14	Reserved
15 DRMD	Data Rate Controller mode. Defines the mode setting for LFAST slave device by S/W or LFAST master. 0S/W controls the Data Rate controller mode. In LFAST Slave the ICLC frames for rate change have no effect on the Data rate. 1In LFAST Slave the reception of ICLC frame for rate change sets appropriate speed mode. In LFAST Master the SCR[DRMD] should be 0.
16 - 22	Reserved
23 RDR	Receiver Data Rate. This bit defines the receiver data rate. For LFAST Master: – S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Tx interface. For LFAST Slave: – When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Rx interface. – The S/W can program this bit when SCR[DRMD] = 0. – This bit is cleared on MCR[DRFEN] negation. 0Data rate of Rx block is low speed. 1Data rate of Rx block is high speed.

Table 677. SCR field descriptions(Continued)

Field	Description
24-30	Reserved
31 TDR	Transmit Data Rate. This bit defines the transmitter data rate. For LFAST Master: – The S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Rx interface. For LFAST Slave: – When SCR[DRMD] = 1, the H/W writes this bit on receipt of an ICLC frame changing the speed mode of the slaves Tx interface. – The S/W can program this bit when SCR[DRMD] = 0. – This bit is cleared on MCR[DRFEN] = 0. 0Data rate of Tx block is low speed. 1Data rate of Tx block is high speed.

47.6.2.3 Correlator Control Register (COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Offset: 0008h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMPSEL(1)								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	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	CORRTH(1)			0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

1 Only writable only when MCR[RXEN] = 0.

Figure 655. Correlator Control Register (COCR)

Table 678. COCR field descriptions

Field	Description
0-7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection). Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. 00h Sampler Data Path selected by correlator 01h Sampler Data Path 0 selected 02h Sampler Data Path 1 selected 04h Sampler Data Path 2 selected 08h Sampler Data Path 3 selected 10h Sampler Data Path 4 selected 20h Sampler Data Path 5 selected 40h Sampler Data Path 6 selected others Sampler Data Path 7 selected
8-27	Reserved
28-30 CORRTH	Correlator threshold level. Defines the correlation threshold level. 000 9 Bits of correlation 001 10 Bits of correlation … 110 15 Bits of correlation 111 16 Bits of correlation
31	Reserved

47.6.2.4 Test Mode Control Register (TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes

Offset: 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	CLKTST	LPON	0	0	0	0	0	LPMOD(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	LPFRMTH(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 Writable only when TMCR[LPON] = 0.

Figure 656. Test Mode Control Register (TMCR)

Table 679. TMCR field descriptions

Field	Description
0-5	Reserved
6 CLKTST	Clock Test mode. S/W can define when the clock test mode is enabled. This bit can also be set and cleared by LFAST slave H/W on reception of an ICLC command. 1Clock Test mode enabled 0Clock Test mode disabled
7 LPON	Loopback mode Logic Enable. S/W can define when the loopback logic is enabled. This bit can also be written by LFAST slave H/W on reception of an ICLC command. 1Loopback mode is enabled 0Loopback mode is disabled
8-12	Reserved
13-15 LPMOD	Loopback mode. Defines the type of loopback mode enabled. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation Others - Reserved (should not be written by S/W)
16-31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0000h Reserved.(Not to be used) 0001h Check 1 frame have correct sync, header and payload. … FFFEh Check 65534 frames have correct sync, header and payload. FFFFh Check 65535 frames have correct sync, header and payload.

47.6.2.5 Auto Loopback Control Register (ALCR)

Offset: 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	LPCNTEN(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 W	LPFMCNT(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 Writable only when TMCR[LPON] = 0.

Figure 657. Auto Loopback Control Register (ALCR)

Table 680. ALCR field descriptions

Field	Description
0-14	Reserved
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable. Enables fixed number of auto predefined loopback frame transmission. 0 Infinite predefined loopback frame transmission enabled 1 Fixed count of predefined loopback frame transmission enabled
16-31 LPFMCNT	Auto Loopback Frame Transmission Count. Defines the number of predefined auto loopback frames to be sent. The predefined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0000h Reserved.(Not to be used) 0001h Send 1 frame with correct sync, header and payload … FFFEh Send 65534 frames with correct sync, header and payload FFFFh Send 65535 frames with correct sync, header and payload

47.6.2.6 Rate Change Delay Control Register (RCDCR)

Offset: 0014h 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	DRCNT(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	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 Only writable when MCR[DRFEN] = 1.

Figure 658. Rate Change Delay Control Register (RCDCR)

Table 681. RCDCR field description

Field	Description
0-11	Reserved
12-15 DRCNT	Data Rate Controller Counter Value. Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. 0h 0 cycles 1h 1 cycles … Eh 14 cycles Fh 15 cycles
16-31	Reserved

47.6.2.7 Wakeup Delay Control Register (SLCR)

Offset: 0018h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HSCNT(1)								0	0	0	0	LSCNT(1)			
W																
Reset	0	0	0	1	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	HWKCNT(1)								0	0	0	0	LWKCNT(1)			
W																
Reset	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

1 Only writable when MCR[DRFEN] = 0.

Figure 659. Wakeup Delay Control Register (SLCR)

Table 682. SLCR field descriptions

Field	Description
0-7 HSCNT	High Speed Sleep mode Exit Time. Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. 00h 0 cycle 01h 1 cycle … 12h 18 cycles (200 ns + 8 cycles of High speed clock) … FEh 254 cycles FFh 255 cycles
8-11	Reserved
12-15 LSCNT	Low Speed Sleep mode Exit Time. Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. 0h 0 cycle 1h 1 cycle (200ns + 1 cycle of Low speed clock) … Eh 14 cycles Fh 15 cycles
16-23 HWKCNT	Wake Up time for the LD. Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. 00h0 cycles 01h1 cycle … 5Fh95 cycles (1.18 us) … FFh255 cycles Maximum
24-27	Reserved
28-31 LWKCNT	Wake Up time for the LD. Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. 0h0 cycle 1h1 cycle 2h2 cycles (1.18 us) … Eh14 cycles Fh15 cycles

47.6.2.8 ICLC Control Register (ICR)

Offset: 001Ch 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	ICLCSEQ	SNDICLC(2)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	ICLCPLD(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 Writable only when ICR[SNDICLC] = 0

2 Set by user software and cleared by system hardware.

Figure 660. ICLC Control Register (ICR)

Table 683. ICR field descriptions

Field	Description
0-13	Reserved
14 ICLCSEQ	ICLC enabled. This bit should be set whenever the S/W is performing a series of ICLC frame transfers. This bit ensures only ICLC frame transmission request is serviced, other pending frames request are ignored. 1 S/W is performing ICLC frame transfers. Only the ICLC frames will be scheduled during this period. All the other frames will be scheduled only after ICR[ICLCSEQ] = 0. 0 Single ICLC frame transfer
15 SNDICLC	ICLC frame request. This bit is set to initiate the transfer of ICLC frame by LFAST master. This bit should be set (ICR[SNDICLC] = 1) after writing the required ICLC payload to be transmitted in the ICLCPLD field of the register. This bit is self clearing, which will be cleared when ICLC frame transfer is complete. 1 Valid ICLC frame for transfer. 0 No Valid ICLC frame for transfer.
16-23	Reserved
24-31 ICLCPLD	ICLC Payload. This field is used to program the payload of the ICLC frame to be transmitted. New ICLC payload should be set when ICR[SNDICLC] = 0. Refer to *Table 707* for supported ICLC payloads.

47.6.2.9 Ping Control Register (PICR)

Offset: 0020h 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	PNGREQ(2)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PNGAUTO(1)	0	0	0	0	0	0	0	PNGPYLD(1)							
W																
Reset	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0

1 Only writable when MCR[DRFEN] = 0.

2 Set by user software and cleared by system hardware.

Figure 661. Ping Control Register (PICR)

Table 684. PICR field descriptions

Field	Description
0-14	Reserved
15 PNGREQ	Ping Response Frame Request. This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. 1 Ping response frame transmission request is queued 0 No pending Ping response frame transmission request
16 PNGAUTO	Ping Response Enable. Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master 0 Ping response should not be automatically sent. 1 Ping response should be automatically sent.
17-23	Reserved
24-31 PNGPYLD	LFAST Slave: Defines the LFAST slaves ping reply frame payload content. LFAST Master: Defines the expected payload of ping response frame. Used for comparison with ping frame received.

47.6.2.10 Rx FIFO CTS Control Register (RFCR)

Offset: 002Ch 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	RCTSMX(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	RCTSMN(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

1 Only writable when MCR[DRFEN] = 0.

Figure 662. Rx FIFO CTS Control Register (RFCR)

Table 685. RFCR field descriptions

Field	Description
0-9	Reserved
10-15 RCTSMX	Rx FIFO Maximum Threshold. Defines the condition for CTS bit of frames to be negated.
16-25	Reserved
26-31 RCTSMN	Rx FIFO Minimum Threshold. Defines the condition for CTS bit of frames to be set.

47.6.2.11 Tx Interrupt Enable Register (TIER)

Offset: 0030h 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	TXIIE	TXOVIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	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	TXPNGIE	0	TXUNSIE	TXICLCIE	TXDTIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 663. Tx Interrupt Enable Register (TIER)

Table 686. TIER field descriptions

Field	Description
0-13	Reserved
14 TXIIE	Tx Data Interface Not Enabled — (Mask) Enables or disables the interrupt. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt is disabled 1 Interrupt is enabled
15 TXOVIE	Transmit Data FIFO Overflow Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
16-26	Reserved
27 TXPNGIE	Ping Response Frame Transmitted Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28	Reserved
29 TXUNSIE	Unsolicited Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
30 TXICLCIE	ICLC Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
31 TXDTIE	Data Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled

47.6.2.12 Rx Interrupt Enable Register (RIER)

Offset: 0034h 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	RXUOIE	RXMNIE	RXMXIE	RXUFIE	RXOFIE	RXSZIE	RXICIE	RXLCEIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	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	RXCTSIE	RXDIE	RXUNSIE	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 664. Rx Interrupt Enable Register (RIER)

Table 687. RIER field descriptions

Field	Description
0-7	Reserved
8 RXUOIE	Unsolicited frame register overflow. Indicates existing unsolicited frame has not been read and a new unsolicited frame has arrived. 0 Interrupt is disabled 1 Interrupt is enabled
9 RXMNIE	Rx Data FIFO Min Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
10 RXMXIE	Rx Data FIFO Max Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
11 RXUFIE	Rx Data FIFO Underflow 0 Interrupt is disabled 1 Interrupt is enabled
12 RXOFIE	Rx Data FIFO Overflow 0 Interrupt is disabled 1 Interrupt is enabled
13 RXSZIE	Frame with unsupported frame size received. Valid frame sizes are defined in *Table 711*. 0 Interrupt is disabled 1 Interrupt is enabled
14 RXICIE	Invalid ICLC code Received 0 Interrupt is disabled 1 Interrupt is enabled
15 RXLCEIE	Invalid Logical Channel Type 0 Interrupt is disabled 1 Interrupt is enabled
16-27	Reserved
28 RXCTSIE	Frame with CTS bit Low Received 0 Interrupt is disabled 1 Interrupt is enabled
29 RXDIE	Data frame received 0 Interrupt is disabled 1 Interrupt is enabled
30 RXUNSIE	Unsolicited Frame received 0 Interrupt is disabled 1 Interrupt is enabled
31	Reserved

47.6.2.13 Rx ICLC Interrupt Enable Register (RIIER)

Offset: 0038h 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														
W			ICPFIE	ICPSIE	ICPRIE	ICTOIE	ICLPIE	ICCTIE	ICTDIE	ICTEIE	ICRFIE	ICRSIE	ICTFIE	ICTSIE	ICPOFIE	ICPONIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 665. Rx ICLC Interrupt Enable Register (RIIER)

Table 688. RIIER field descriptions

Field	Description
0-17	Reserved
18 ICPFIE	Ping Frame Response failed 0 Interrupt is disabled 1 Interrupt is enabled
19 ICPSIE	Ping Frame Response successful 0 Interrupt is disabled 1 Interrupt is enabled
20 ICPRIE	ICLC frame for Ping Frame Request received 0Interrupt is disabled 1 Interrupt is enabled
21 ICTOIE	ICLC frame for Test mode off received 0 Interrupt is disabled 1 Interrupt is enabled
22 ICLPIE	ICLC frame for Loopback On received 0 Interrupt is disabled 1 Interrupt is enabled
23 ICCTIE	ICLC frame for Clk Test mode on received 0 Interrupt is disabled 1 Interrupt is enabled
24 ICTDIE	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt is disabled 1 Interrupt is enabled
25 ICTEIE	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt is disabled 1 Interrupt is enabled
26 ICRFIE	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled

Table 688. RIIER field descriptions(Continued)

Field	Description
27 ICRSIE	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
28 ICTFIE	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
29 ICTSIE	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
30 ICPOFIE	ICLC frame for PLL OFF received 0 Interrupt is disabled 1 Interrupt is enabled
31 ICPONIE	ICLC frame for PLL ON received 0 Interrupt is disabled 1 Interrupt is enabled

47.6.2.14 LVDS Control Register (LCR)

Offset: 0040h 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	SWWKLD(1)	SWSLPLD(1)	SWWKLR(1)	SWSLPLR(1)	SWOFFLD(1)	SWONLD(1)	SWOFFLR(1)	SWONLR(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	LVRXOFF	LVTXOE	TXCMUX	LVRFEN	LVLPEN	0	0	0	0	0	LVRXOP_TR	Reserved	LVRXOP_BR	LVTXOP	LVCKSS	LVCKP
W																
Reset	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0

1 Set by user software and cleared by system hardware.

Figure 666. LVDS Control Register (LCR)

Table 689. LCR field description

Field	Description
0-7	Reserved
8 SWWKLD	SW signal to take LVDS LD out of Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep)

Table 689. LCR field description(Continued)

Field	Description
9 SWSLPLD	SW signal to put LVDS LD into Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be put in sleep
10 SWWKLR	SW signal to take LVDS LR out of Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep)
11 SWSLPLR	SW signal to put LVDS LR into Sleep mode. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up)
12 SWOFFLD	SW signal to turn OFF the LVDS LD. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be turned OFF (provided no other source is trying to turn the LD ON)
13 SWONLD	SW signal to turn ON the LVDS LD. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be turned ON
14 SWOFFLR	SW signal to turn OFF the LVDS LR. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be turned OFF (provided no other source is trying to turn the LR ON)
15 SWONLR	SW signal to turn ON the LVDS LR. This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be turned ON
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode.
17 LVTXOE	LVDS LD output buffer enable. 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux. The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad
19 LVRFEN	LVDS pad reference enable. 0 LVDS reference pad disabled 1 LVDS reference pad enabled
20 LVLPEN	Tx LVDS internal loopback enable. The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
19-25	Reserved
26 LVRXOP_TR	Used to enable or disable the on-chip receiver termination resistor in LFAST mode (applies to LVDS pad use for LFAST only): 1Disable on-chip LFAST receiver termination 0Enable on-chip LFAST receiver termination
27	Reserved

Table 689. LCR field description(Continued)

Field	Description
28 LVRXOP_BR	Used to set the bias current for the receiver in LFAST mode. It is recommended to always write 1 to this bit when using the LFAST interface. Writing 0 will allow for a small power savings during lower baud rates (applies to LVDS pad use for LFAST only): 0 Use for LFAST receiver baud rates less than maximum baud rate 1 Required for LFAST receiver maximum baud rate
29 LVTXOP	Control signal for LFAST and Micro-Second Bus selection. 0 Micro-Second Bus selection 1 LFAST Bus selection
30 LVCKSS	LVDS Clock Sync Select. This bit is used to adjust the LVDS data sampling to the duty cycle of the clock. It is recommended that a value of zero is used in all cases. A value of one is reserved for specific devices/revisions with different clock timing. 0 normal clock used to sample the LVDS data 1 adjusted clock used to sample the LVDS data
31 LVCKP	LVDS clock select. This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

47.6.2.15 Unsolicited Tx Control Register (UNSTCR)

Offset: 0044h 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	USNDRQ(2)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	UNSHDR(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 Only writable when UNSTCR[USNDRQ] = 0

2 Set by user software and cleared by system hardware.

Figure 667. Unsolicited Tx Control Register (UNSTCR)

Table 690. UNSTCR field descriptions

Field	Description
0-14	Reserved
15 USNDRQ	Tx Unsolicited send request. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission

Table 690. UNSTCR field descriptions(Continued)

Field	Description
16-24	Reserved
25-31 UNSHDR	Tx Unsolicited message header.

47.6.2.16 LFAST Unsolicited Tx Data Registers (UNSTDR0–UNSTDR8)

Offset: 0048h (UNSTDR8)....0068h (UNSTDR0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	UNTXD[31:16](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	UNTXD[15:0](1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 Only writable when UNSTCR[USNDRQ] = 0

Figure 668. Unsolicited Tx Data Registers (UNSTDR0–UNSTDR8)

Table 691. UNSTDR*n* field descriptions

Field	Description
0-31 UNTXD	Unsolicited Transmit Data 8-0. This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

47.6.2.17 Global Status Register (GSR)

Offset: 0080h Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUALMD	0	0	0	0	0	0	0	0	0	0	0	0	LRMD	LDSM	DRSM
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	LPTXDN	LPFPDV	LPCPDV	LPCHDV	LPCSDV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 669. Global Status Register (GSR)

Table 692. GSR field descriptions

Field	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module Slave only mode 1 LFAST Module in dual mode
1-12	Reserved
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active. 0 Rx Controller is in Idle state 1 Rx Controller is active
14 LDSM	Transmit Interface Data Rate Status. Indicates the current speed rate of the Tx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Receive Interface Data Rate Status. Indicates the current speed rate of the Rx controller. 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
16-26	Reserved
27 LPTXDN	Auto loopback frame transmission count reached. The Count of frame is defined by Auto Loopback Control Register (ALCR) LPFMCNT. 0 Auto loopback frame transmission count not reached. 1 Auto loopback frame transmission count reached.
28 LPFPDV	Loopback frame pass threshold reached. 0 Pass frame threshold not reached. 1 Pass frame threshold achieved
29 LPCPDV	Valid payload received during loopback check mode. Indicates whether the Loop back frame received payload is CBh. 0 Payload received is not CBh 1 Payload received is CBh

Table 692. GSR field descriptions(Continued)

Field	Description
30 LPCHDV	Valid header received during loopback check mode. Indicates whether the Loop back frame received header is 13h. 0 Header received is not 13h 1 Header received is 13h
31 LPCSDV	Valid synchronization received. 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

47.6.2.18 Ping Status Register (PISR)

Offset: 0084h 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	RXPNGD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 670. Ping Status Register (PISR)

Table 693. PISR field descriptions

Field	Description
0-23	Reserved
24-31 RXPNGD	Ping Data Register. In LFAST Master mode the Ping response ICLC frame received is stored into this register.

47.6.2.19 Data frame Status Register (DFSR)

Offset: 0094h Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RXDCNT						0	0	0	0	0	RXFCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			TXDCNT						0	0	0	0	0	TXFCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 671. Data frame Status Register (DFSR)

Table 694. DFSR field descriptions

Field	Description
0-1	Reserved
2-7 RXDCNT	Unread Rx Frame Data Count. Indicates the number of unread data stored in the Rx Data FIFO.
8-12	Reserved
13-15 RXFCNT	Unread Rx Frame Count. Indicates the number of unread data frames stored in the Rx Data FIFO.
16-17	Reserved
18-23 TXDCNT	Unread Tx Frame Data Count. Indicates the number of unread data stored in the Tx Data FIFO.
24-28	Reserved
29-31 TXFCNT	Unread Tx Frame Count.Count of pending Data Frames programmed by System Side Module.

47.6.2.20 Tx Interrupt Status Register (TISR)

Offset: 0098h 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	TXIEF	TXOVF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TXPNGF	0	TXUNSF	TXICLCF	TXDTF
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 672. Tx Interrupt Status Register (TISR)

Table 695. TISR field descriptions

Field	Description
0-13	Reserved
14 TXIEF	TxData Interface not enabled. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table 695. TISR field descriptions(Continued)

Field	Description
16-26	Reserved
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
28	Reserved
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

47.6.2.21 Rx Interrupt Status Register (RISR)

Offset: 009Ch 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	RXUOF	RXMNF	RXMXF	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
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

	16	17	18	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	RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 673. Rx Interrupt Status Register (RISR)

Table 696. RISR field descriptions

Field	Description
0-7	Reserved
8 RXUOF	Unsolicited frame register overflow. Indicates existing unsolicited frame has not been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table 696. RISR field descriptions(Continued)

Field	Description
9 RXMNF	Rx Data FIFO Min Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
10 RXMXF	Rx Data FIFO Max Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
11 RXUFF	Rx Data FIFO Underflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
12 RXOFF	Rx Data FIFO Overflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
13 RXSZF	Frame with unsupported frame size received. 0 Interrupt event has not occurred 1 Interrupt event has occurred- On reception of frame with payload size for a frame other than mentioned in *Table 711*.
14 RXICF	Invalid ICLC code Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 RXLCEF	Invalid Logical Channel Type. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame other than mentioned in *Table 711*.
16-27	Reserved
28 RXCTSF	Frame with CTS bit Low Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 RXDF	Data frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 RXUNSF	Unsolicited Frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
31	Reserved

47.6.2.22 Rx ICLC Interrupt Status Register (RIISR)

Offset: 00A0h 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	ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 674. Rx ICLC Interrupt Status Register (RIISR)

Table 697. RIISR field descriptions

Field	Description
0-17	Reserved
18 ICPFF	Ping Frame Response failed 0 Interrupt event has not occurred 1 Interrupt event has occurred
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred 1 Interrupt event has occurred
20 ICPRF	ICLC Ping Frame Request received 0 Interrupt event has not occurred 1 Interrupt event has occurred
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred 1 Interrupt event has occurred
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred 1 Interrupt event has occurred
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred 1 Interrupt event has occurred
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table 697. RIISR field descriptions(Continued)

Field	Description
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred 1 Interrupt event has occurred

47.6.2.23 PLL and LVDS Status Register (PLLLSR)

Offset: 00A4h 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	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	0	0	0	0	LRSLPS	LDSLPS	LDPDS	LRPDS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 675. PLL and LVDS Status Register (PLLLSR)

Table 698. PLLLSR field descriptions

Field	Description
0-27	Reserved
28 LRSLPS	This bit indicates the real time status of the LR sleep signal. 0 LR sleep signal is negated. 1 LR sleep signal is asserted.
29 LDSLPS	This bit indicates the real time status of LD sleep signal. 0 LD power sleep signal is negated. 1 LD power sleep signal is asserted.

Table 698. PLLLSR field descriptions(Continued)

Field	Description
30 LDPDS	This bit indicates the real time status of LD power down signal. When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LR power down signal. When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.

47.6.2.24 Unsolicited Rx Status Register (UNSRSR)

Offset: 00A8h 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	URXDV	0	0	0	0	0	URPCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 676. Unsolicited Rx Status Register (UNSRSR)

Table 699. UNSRSR field descriptions

Field	Description
0-22	Reserved
23 URXDV	Unsolicited data valid. Indicates a valid frame exists in the Unsolicited Data registers.
24-28	Reserved
29-31 URPCNT	Rx Unsolicited payload. Indicates the number of bytes of valid Rx unsolicited data payload present in the Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0). The bitfield coding is same as b7-b5 in *Table 673*.

47.6.2.25 Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)

Offset: 00ACh (UNSRDR8)....00CCh (UNSRDR0) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UNRXD[31:16]															
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	UNRXD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 677. Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)

Table 700. UNSRDRn field descriptions

Field	Description
0-31 UNRXD	Unsolicited Receive Data. This represents 9 registers for Unsolicited received data.It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288 bit payload.

47.6.3 Register safety classification requirements

This module is classified as NoSaMo (Non-Safety relevant Module) for safety requirements[o]. All registers of the LFAST module are classified as non-safety relevant.

The MCR[IPGDBG] fulfills the safety requirements for monitoring of the debug freeze signal.

47.7 Functional description

47.7.1 Startup procedure

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

o. See the “Functional Safety” chapter for register classification details.

47.7.1.1 LFAST master interface startup procedure

1. After reset the SLCR and RCDCR are programmed according to the LVDS parameters in the device datasheet.
2. The LCR is programmed with configuration parameters of the LVDS.
3. Write MCR[MSEN] = 1. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
4. Write MCR[DRFEN] = 1 to enable the LFAST.
5. Write MCR[RXEN] = 1 and MCR[TXEN] = 1 to negate the LD powerdown, LR disable and LR powerdown signals.
6. Write ICR[ICLCPLD] = 31h to enable the Slaves Tx Interface.
7. Write ICR[SNDICLC] = 1 and ICR[ICLCSEQ] = 1.
8. Write MCR[TXARBD] = 0.
9. The ICLC transmission is confirmed by verifying one of the following:
   – ICR[SNDICLC] = 0.
   – TISR[TXICLCF] = 1.
10. Write ICR[ICLCPLD] = 00h to check the LFAST slaves status.
11. Write ICR[SNDICLC] = 1.
12. The LFAST slave status is confirmed by occurrence of one of the following:
   – LFAST slave is enabled if RIISR[ICPSF] = 1. Proceed to step 14.
   – LFAST slave is disabled if RIISR[ICPFF] = 1. The LFAST master must wait and restart from Step 7.

Speed mode change:

13. Write LCR[SWPON] = 1 to enable the LFAST masters PLL.
14. Write ICR[SNDICLC] = 1.
15. The ICLC transmission is confirmed by occurrence of one of the following:
   – ICR[SNDICLC] = 0.
   – TISR[TXICLCF] = 1.
16. To change LFAST masters Tx interface speed both of the following operations are performed:
   Write ICR[ICLCPLD] = 10h for Tx data fast frame.
   – Write ICR[SNDICLC] = 1.
17. Both of the following operations are performed to change the LFAST masters Rx interface speed:
   – Write ICR[ICLCPLD] = 80h to select Rx data fast frame.
   – Write ICR[SNDICLC] = 1.

Note: The slaves Tx interface speed mode should be changed first

18. The ICLC transmission is confirmed by the occurrence of one of the following:
    – ICR[SNDICLC] = 0
    – TISR[TXICLCF] = 1.
19. Write SCR[TDR] = 1 to change the LFAST masters Tx interface speed.
20. Write SCR[RDR] = 1 to change the LFAST masters Rx interface speed.
21. Write ICR[ICLCPLD] = 00h to confirm the change in speed of the LFAST slave. This frame should be written after waiting for the expected delay in the start of the PLL and speed mode change delay at the LFAST slave.
22. Write ICR[SNDICLC] = 1
23. Write MCR[TXARBD] = 1, after some delay to ensure the step 24 ICLC frame is send but no other frame is sent to arbitation.
24. The LFAST slaves speed mode is confirmed by the occurrence of the following:
    – If RIISR[ICPSF] = 1. The LFAST slave is in High Speed mode.
    – If RIISR[ICPFF] = 1. The LFAST slave is in Low Speed mode.
25. If RIISR[ICPSF] = 1, then all frame arbitration is enabled by both of the following operations:
    – Write ICR[ICLCSEQ] = 0.
    – Write MCR[TXARBD] = 0.
26. If RIISR[ICPFF] = 1 modifying the LFAST slaves speed needs to be repeated by:
    – Write SCR[TDR] = 0 to return the Tx interface to Low Speed mode.
    – Write MCR[TXARBD] = 0, to enable frame transmission.
    – The operations from step 14 need to be executed again.

47.7.1.2 LFAST slave interface startup procedure

1. After reset the SLCR and RCDCR are programmed according to the LVDS parameters in the device datasheet.
2. The LCR is programmed with the configuration parameters for the LVDS.
3. Write MCR[MSEN] = 0. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
4. Write MCR[DRFEN] = 1 to enable the LFAST.
5. The LR is enabled by writing MCR[RXEN] = 1. This negates the LR disable signal and LR's power down signal.
6. The LD enable signal needs to be asserted. This is done when one of the following are true:
    – When an ICLC frame with payload 31h is received the H/W will write RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
    – MCRMCR[TXEN] = 1
7. After a write to MCR[TXARBD] = 0, a ping frame will be sent on one of the following conditions:
    – When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h frame is received.H/W writes [ICRPF] = 1
    – PICR[PNGREQ] = 1.

Speed mode change:

8. The speed of the Tx interface is changed on one of the following conditions:
   – When SCR[DRMD] = 1 and an ICLC frame with payload 80h is received. H/W writes RIISR[ICTFF] = 1 and SCR[TDR] = 1.
   – SCR[TDR] = 1 when SCR[DRMD] = 0
9. The speed of the Rx interface is changed on one of the following conditions:
   – When SCR[DRMD] = 1 and an ICLC frame with payload 10h is received. H/W writes RIISR[ICRFF] = 1 and SCR[RDR] = 1.
   – SCR[RDR] = 1, when SCR[DRMD] = 0
10. A Ping Frame is sent on one of the following conditions:
   – When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h is received.H/W writes RIISR[ICRPF] = 1
   – Write PICR[PNGREQ] = 1.

47.7.2 Line Receiver (LR)

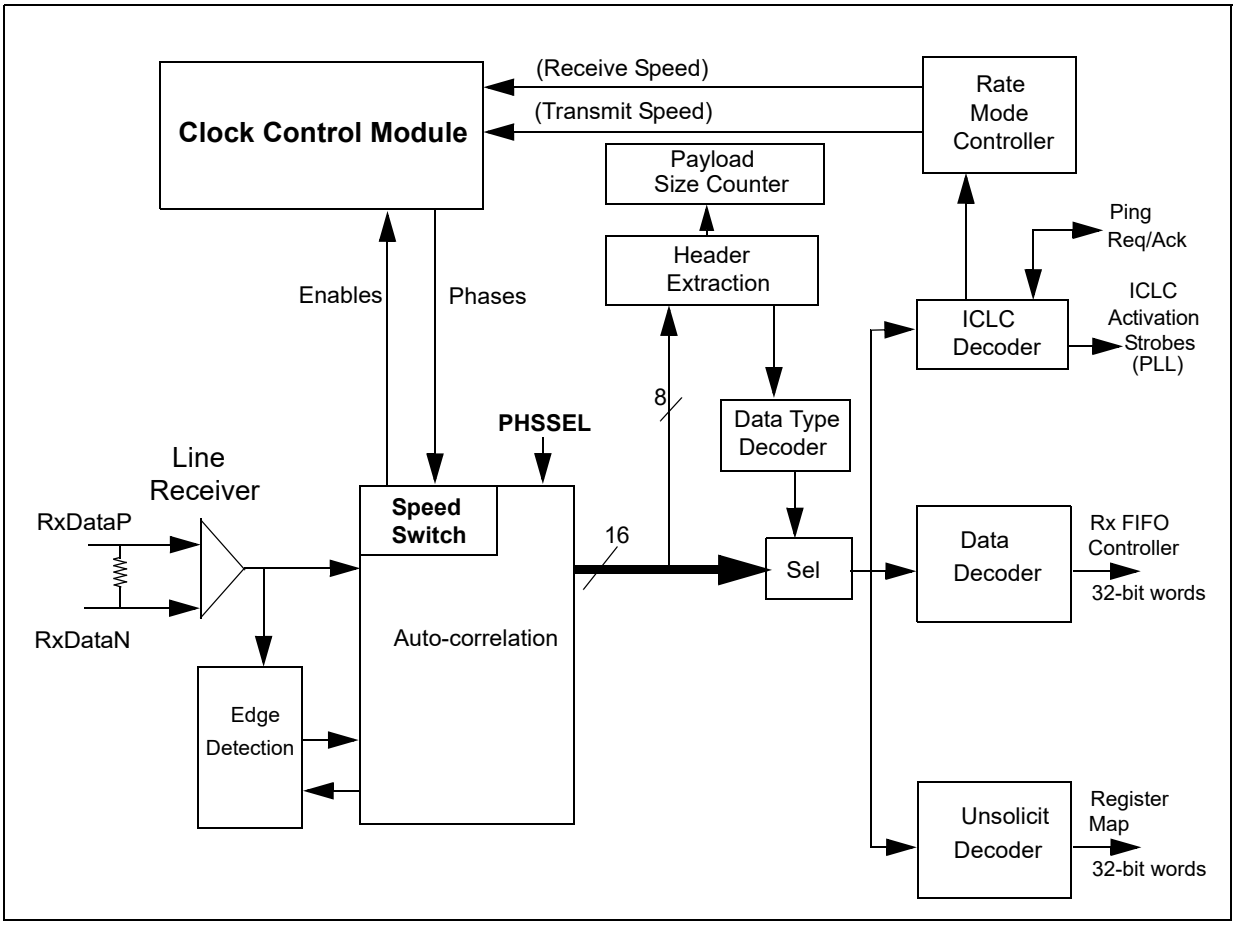
47.7.2.1 Introduction

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto-correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 678 shows the block diagram of Uplink Controller.

Figure 678. Top Level Receive Controller



47.7.2.2 Edge-Detection & Auto-correlation

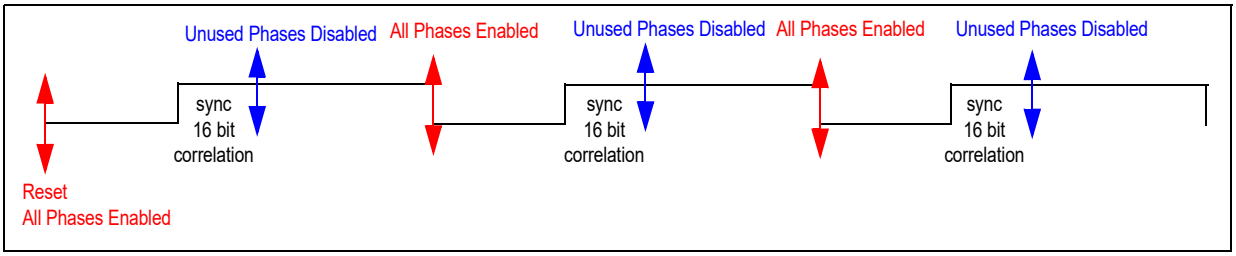
The edge-detection and auto-correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge-detection circuitry is used or not. The edge-detection will be performed first if required before auto-correlation.

47.7.2.2.1 Auto-correlation modes

47.7.2.2.1.1 Hunt correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 679. Hunt Correlation mode



47.7.2.2.2 Edge detection

The edge detector is disabled in Hunt Correlation mode.

47.7.2.2.3 Auto-correlation

The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto-correlation (or synchronization) scheme is to estimate the best clock phase.

The objective of the auto-correlation is to select the clock phase that occurs closest to the center of the eye diagram window.

For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in *Figure 680* and *Figure 681*.

Figure 680. High Speed 4 Phase Clock Alignment Example

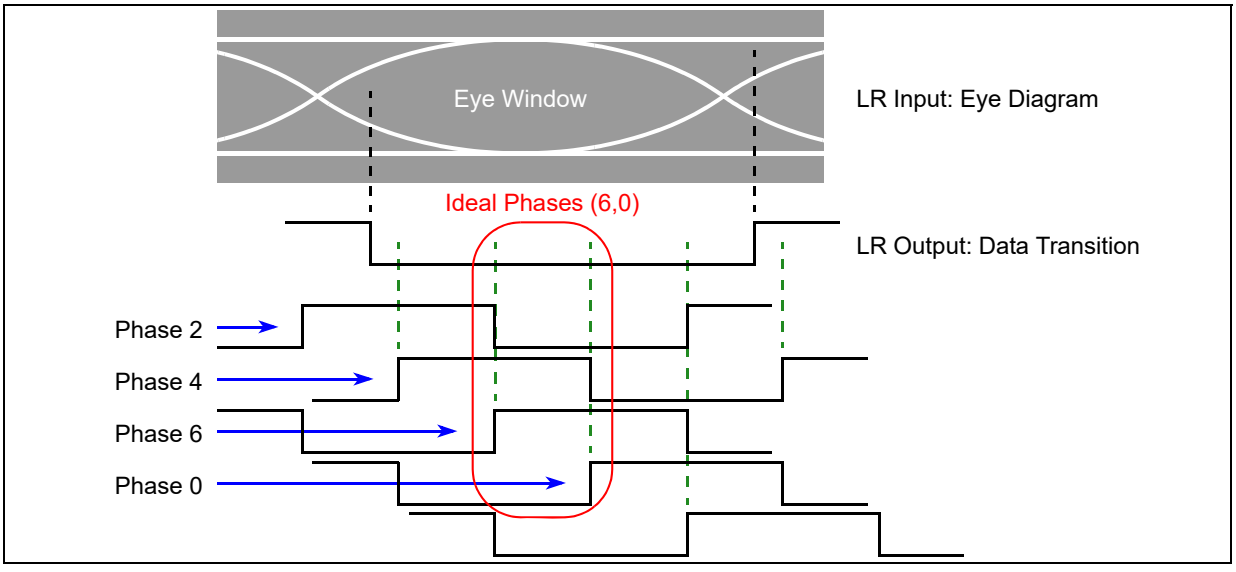
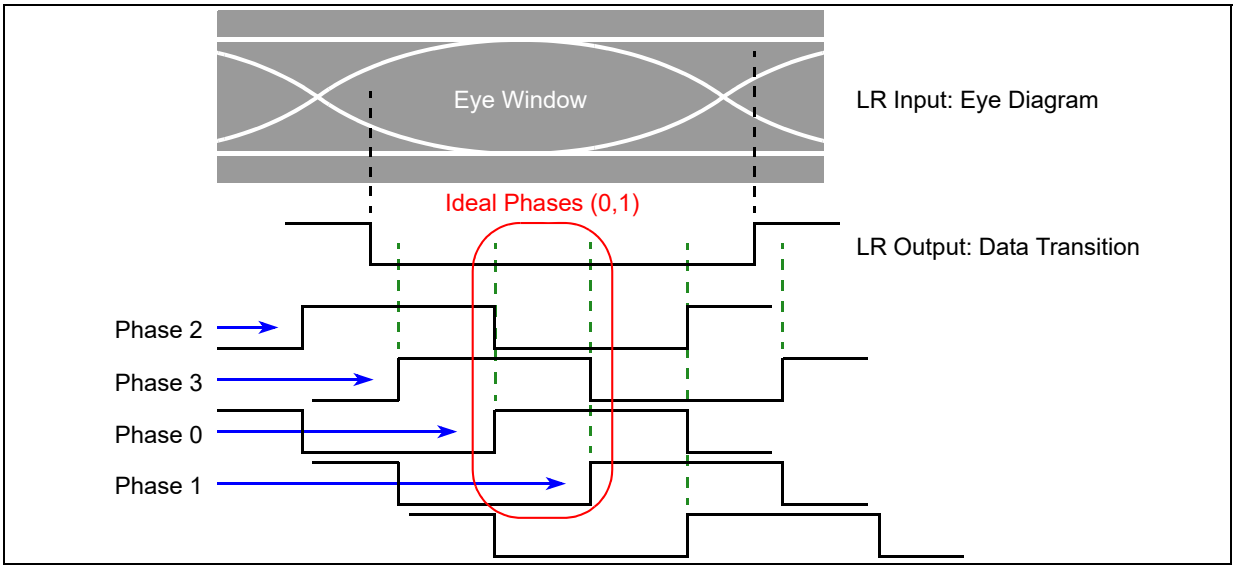


Figure 681. Low speed 4 phase clock alignment example



The auto-correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High Speed 4 Phases: Samplers 0,2,4,6
- Low Speed 4 Phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and re-samples it to a intermediate phase as shown in *Table 701*, and *Table 702* before re-sampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 701. High speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See *Table 701*, Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 702. Low speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See *Table 702*, Samplers 4, 5, 6, 7 are disabled.The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

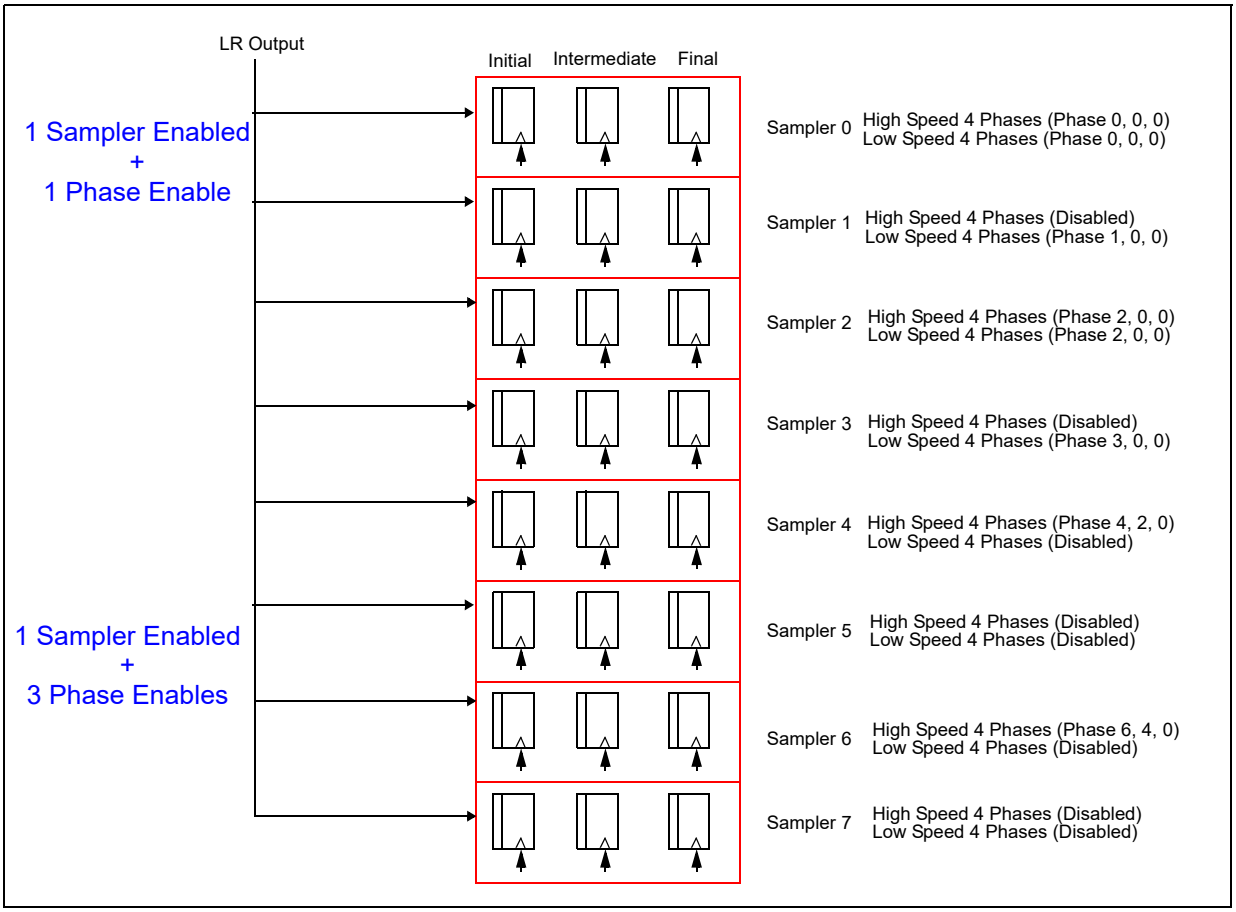
47.7.2.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto-correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in *Figure 682*.

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (e.g Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by de-asserting the respective phase enables.

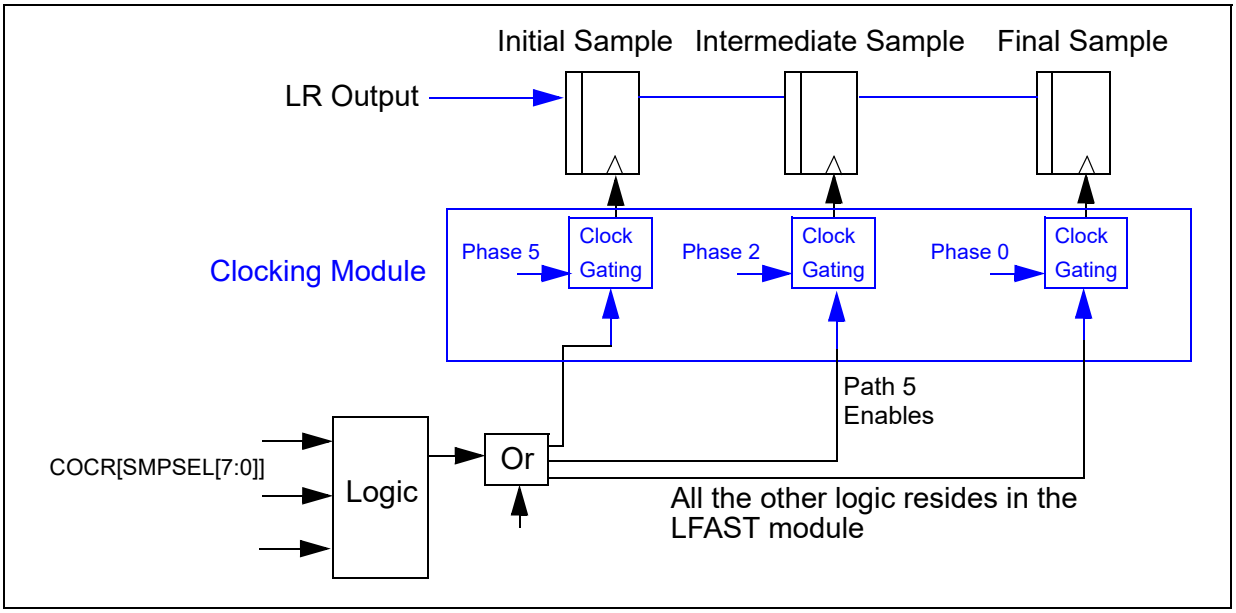
Figure 682. Samplers, each Sampler has 3 Registers



In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in *Figure 683*.

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

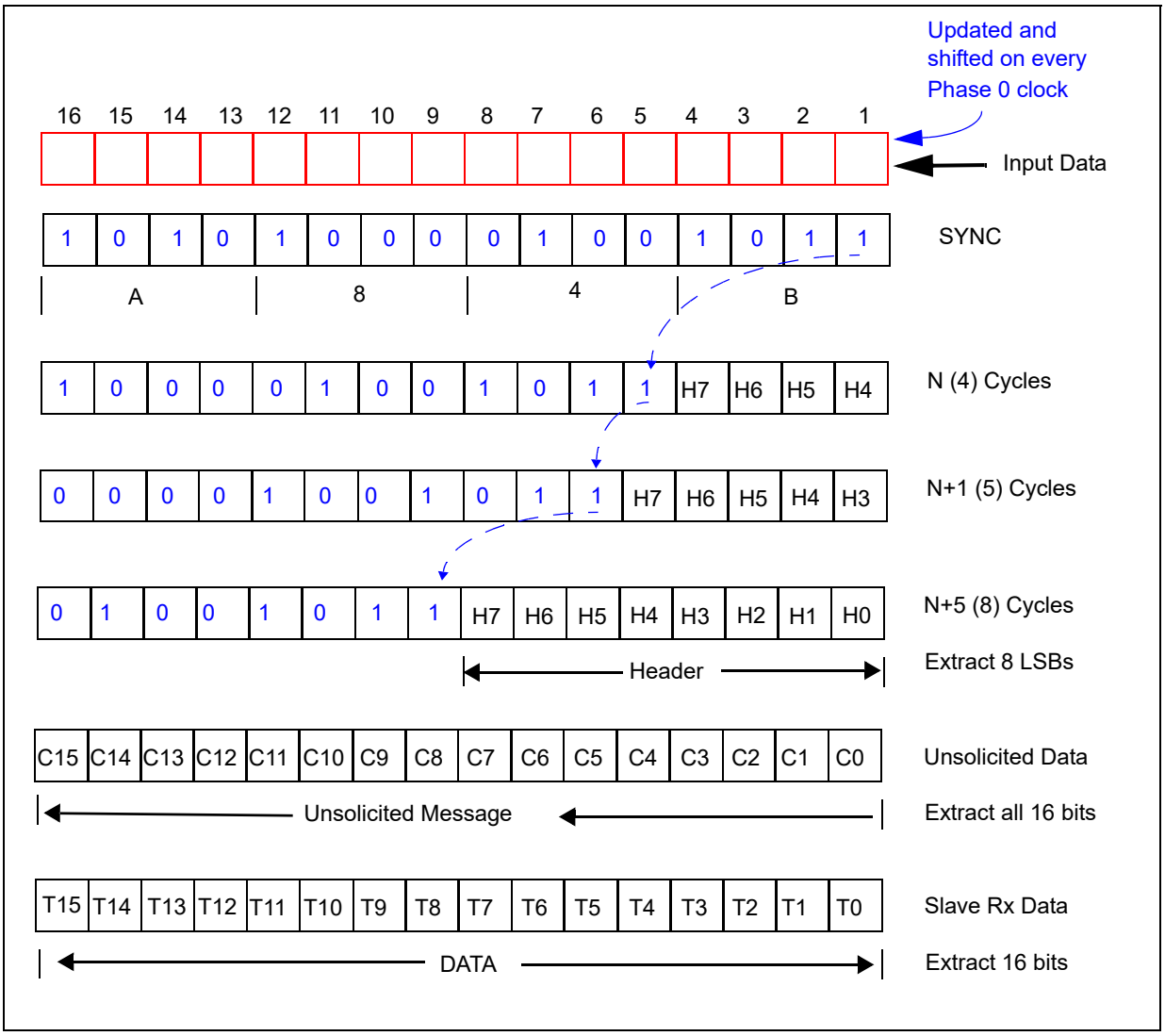
Figure 683. Sampler 5 Logic



47.7.2.3 Header and Payload Extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16-bit word, bit shifted every Phase 0 clock as shown in *Figure 684*.

Figure 684. Extracting Header and Payload from the 16-bit output from Auto-correlation.



47.7.2.3.1 Line Receiver states

The LD has the following states:

- Shutdown
  The LR enters in the shutdown state when the power down signal is high. In this state the LR regulator is supplying a Vdd level, however the LR is not enabled. In this state the LR holds its off value. Once power down signal is negated, a predefined settling time is required before the LR may be used for communication. During the settling time the peer LFAST device should not start its transfer.
- Sleep
  The LR enters in the Sleep state when the sleep signal is high. In the sleep state, the LR is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In this

mode the receiver current consumption is reduced. Hysteresis in the LR ensures that it continues to present a 1 to the LFAST module.

- Normal
  In the Normal state, the LR is primed for reception. Receiver converts differential signals to a single ended logic signal.

Table 703. Line Receiver State

LR State	LFAST Master Triggers (at least one of the following)	LFAST Slave Triggers (at least one of the following)
Shutdown	– Programing of LCR[SWOFFLR] or LCR{SWONLR] – Programing MCR[RXEN] and MCR[TXEN]	– LFAST interface enable assertion/negation – Programing of LCR[SWOFFLR] or LCR{SWONLR] – Programing MCR[RXEN] and MCR[TXEN]
Sleep	– Programing of LCR[SWSLPLR] or LCR[SWWKLR]	– Programing of LCR[SWSLPLR] or LCR[SWWKLR]
Normal	Absence of above mentioned conditions	Absence of above mentioned conditions

47.7.3 Transmit Controller

47.7.3.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, unsolicited, ICLC, CTS, ping etc. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Receive Controller and the ping response from the register block.

The Tx interface has two speed modes:

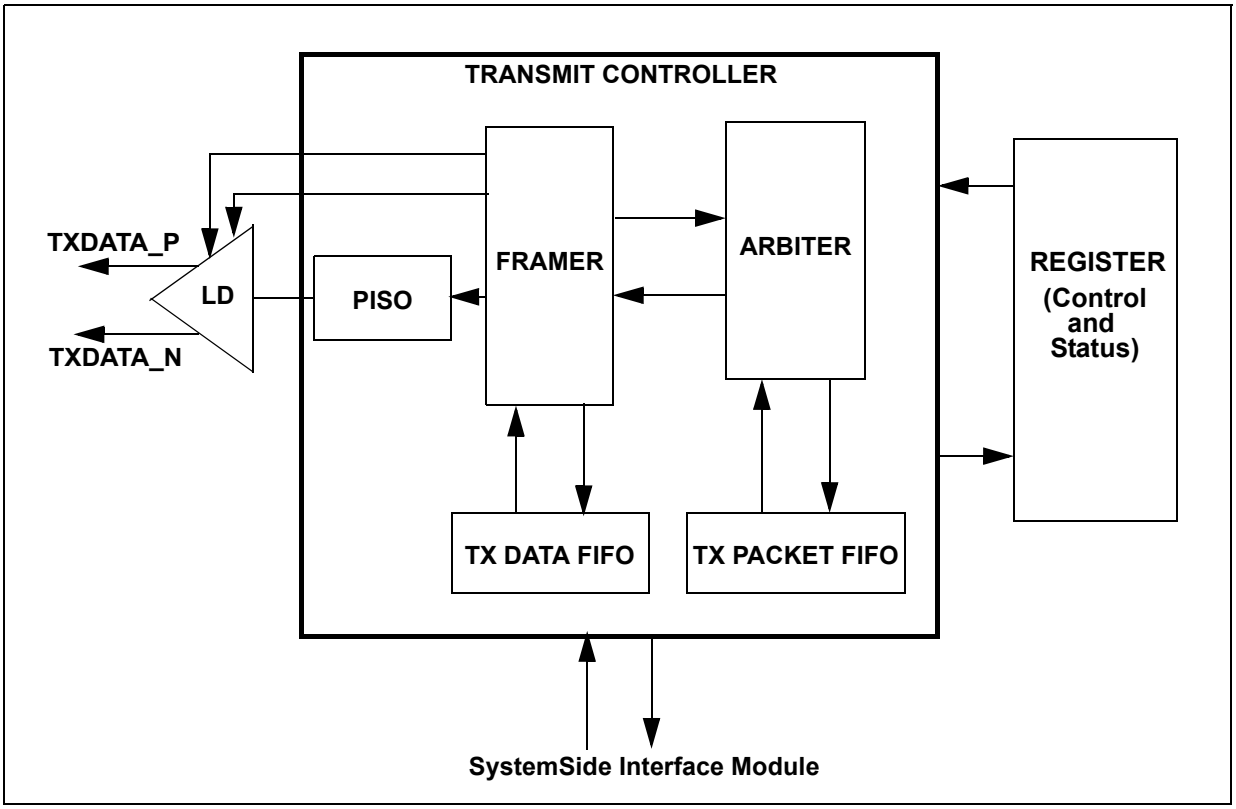
- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in *Figure 685*.

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

Figure 685. Transmit Controller Connections



47.7.3.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like:

- S/W programmable registers
- System side interface FIFO
- Rx interface controller

The level of priority for each request is shown in *Table 704*.

Table 704. Priority Levels for the Transmit Controller

Request	LFAST Master Priority	LFAST Slave Priority	Triggering Conditions (at least one of the listed)
LFAST interface enable	1	1	– LFAST interface enable is asserted – LFAST interface enable is negated
Tx Interface disabled	2	2	– MCR[DRFEN] = 0 – MCR[TXEN] = 0 – MCR[DRFRST] = 1 – MCR[TXARBD] = 1 – LFAST Slave: ICLC frame with payload for “Disable Rx interface” received
Tx Interface speed mode change	3	3	– SCR[TDR] is modified – LFAST Slave: ICLC frame with payload for changing Rx interface speed received

Table 704. Priority Levels for the Transmit Controller(Continued)

Request	LFAST Master Priority	LFAST Slave Priority	Triggering Conditions (at least one of the listed)
Loopback frame in Loopback mode	4	4	– TMCR[LPON] = 1 – LFAST Slave: ICLC frame with payload for loopback mode enable received **Note:** Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
ICLC frame request	5	–	– LFAST Master: ICR[SNDICLC] = 1 – LFAST Master: ICR[ICLCSEQ] = 1
Ping response request	–	5	– LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 – LFAST Slave: PICR[PNGREQ] = 1
Unsolicited frame request	6	6	– Last Frame with CTS = 1 received from the peer LFAST device – UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	7	– Tx FIFO contains one or more frames – Last Frame with CTS = 1 received from the peer LFAST device – MCR[DATAEN] = 1
CTS Frame	8	8	– MCR[CTSEN] = 1 – Rx FIFO reaches High/Low threshold, defined by TISR and no frame pending
Clock mode test	9	9	– TMCR[CLKTST] = 0 – LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCR[DRCNT].

47.7.3.3 Line Driver digital connections

47.7.3.3.1 Line Driver states

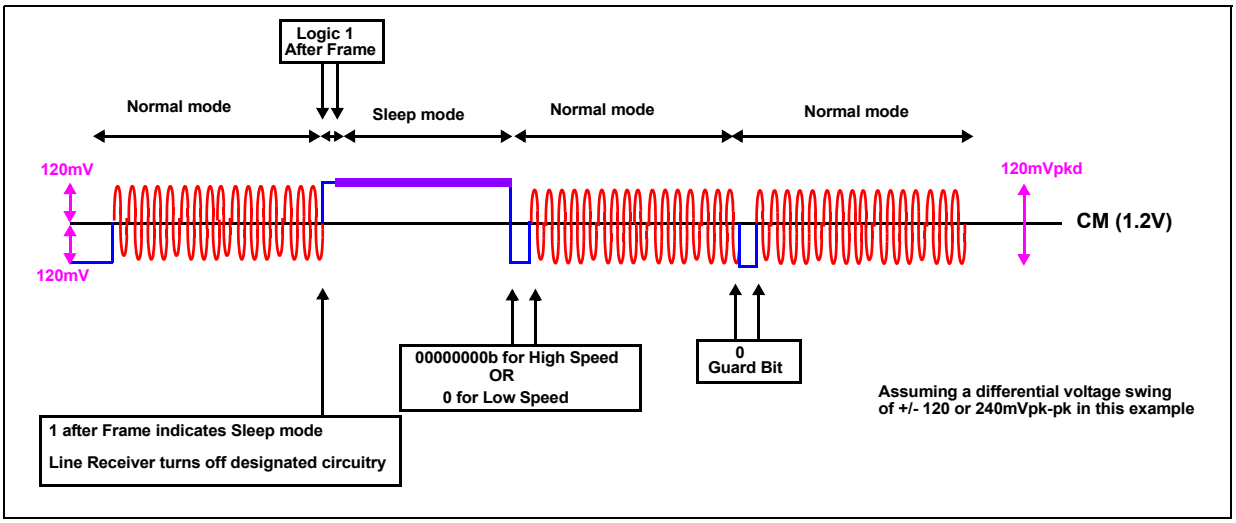
The LD has the following states:

- Shutdown
  The LD enters the Shutdown state when the LD powerdown signal and the output buffer enable is high. In this state, the LD regulator is not supplying power and the LD outputs are connected to ground. Once LD powerdown signal is negated, a settling

time is required before the LD may be used for communication.The settling time is defined by the SLCR[LWKCNT] and SLCR[HWKCNT] bitfields.

- Sleep
  The LD enters in sleep state when the signal is asserted. In this state, the LD is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely.In the sleep state the LD outputs are connected to Vcm (common mode voltage). To exit from Sleep mode the LD sleep signal is negated. The LD is required to transmit a logic 0 level on the interface for a predefined minimum time. The minimum time is the summation of settling time of LD after negation of LD sleep signal and the wakeup time of the LR on the other side. This delay is programmed in the SLCR[HSCNT] and SLCR[LSCNT] bitfields.

Figure 686. Example of Sleep mode



As shown in *Figure 686* before every normal mode burst on the txdatap line there is one guard bit period (minimum inter-frame gap) where a logic 0 will be transmitted.

- Normal
  In the Normal state the LD is primed for transmission. The LD shall drive the interface as dictated by the input data bit that is supplied by a shift register under the control of a finite state machine. If the LD is not enabled, and the framer is activated, and has a frame to transmit, an interrupt (TISR[TXIEF]) will be generated. The LD moves back to the shutdown state when the LFAST master sends an ICLC Rx data off command.

Table 705. Line Driver States

LD State	LFAST Master Triggers	LFAST Slave Triggers
Shutdown	– Programing LVDS[SWOFFLD] and LVDS[SWONLD] – Programing MCR[TXEN] and MCR[DRFEN]	– LFAST interface enable negation/assertion – Programing LVDS[SWOFFLD] and LVDS[SWONLD] – ICLC command from LFAST master – Programing MCR[TXEN] and MCR[DRFEN]
Sleep	– Programing LVDS[SWSLPLD] and LVDS[SWWKLD]	– Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions	Absence of above mentioned conditions

47.7.4 CTS mode support

LFAST module supports flow control method defined in the LFAST3G standard. The CTS (Clear to send) is a protocol defined method to ensure no loss of frame, due to Rx FIFO unavailability. The optimal use of bandwidth of LFAST, without losing frames, is ensured by proper programing of the Rx FIFO Lower threshold (RFCR[CTSMN]) and Higher threshold (RFCR[CTSMX]). The CTS is supported by both the LFAST Master and Slave.

CTS Transmission:

The LFAST device sends CTS information with each frame to the peer LFAST device.

If the CTS mode is enabled by write MCR[CTSEN] = 1 then:

- Sends CTS bit as 0 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the higher threshold defined by the bitfield TISR[CTSMX]. This indicates that the LFAST device does not want the peer device to send data. The CTS bit of all frame are sent 0 until the Rx FIFO pointer reaches the lower threshold, as described below.
- Sends CTS bit as 1 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the Lower threshold defined by the bitfield TISR[CTSMN]. This indicates that the LFAST device is ready to receive data from the peer device. The CTS bit of all frame are sent 1 until the Rx FIFO pointer reaches the Higher threshold, as described above.
- The CTS bit can be sent through any type of frame (data, unsolicited or ICLC). When there are no frames available in LFAST to send to the peer device, a CTS frame is sent. In this frame, bit[4-1] of Header field are sent as 0011b and 8-bit payload of 0.

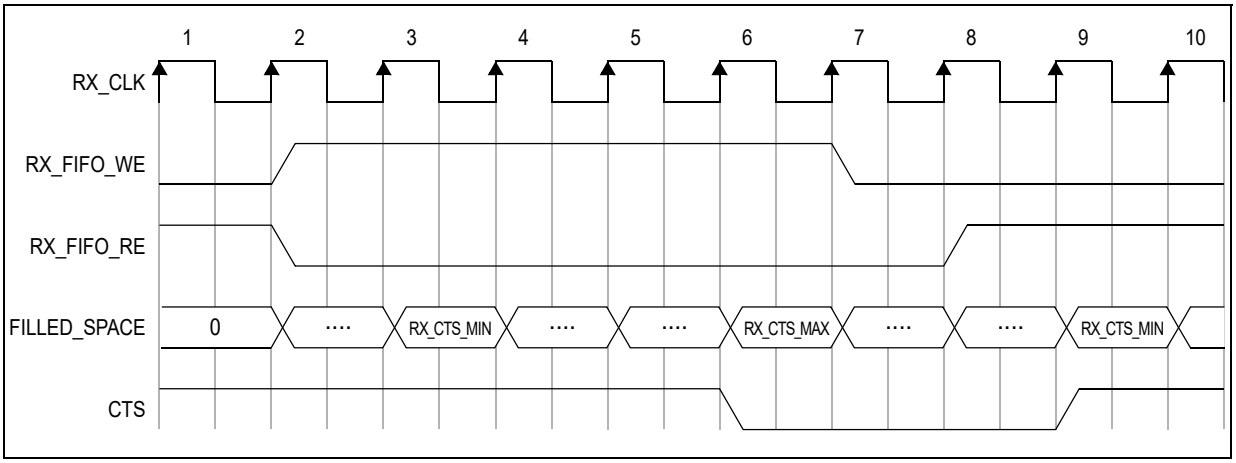
If the CTS mode is not enabled (for example, bit MCR[CTSEN] = 0) then:

- All the frames sent will have CTS bit as 1 in LFAST frame (bit[0] of Header field).

CTS Reception:

The reception of a LFAST frame with CTS bit 0 indicates that the peer device is not ready to receive data frames. The transmission of all pending data and unsolicited frames are postponed until a frame with CTS bit 1 is received.

Figure 687. CTS generation



47.7.5 Frames supported

Table 706 provides the frames supported and their permissible payload sizes

Table 706. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx 2. LFAST Slave Rx	Supported by 1. LFAST Slave Tx 2. LFAST Master Rx	Payload Size (bits)
Data Frame	0100b – 1011b	YES	YES	32, 64, 96, 128, 256, 288
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO(1)	8
CTS Frame	0011b	YES	YES	8
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

If logical channel type (LCT) code other than the above mentioned are received then the interrupt flag RISR[RXLCEF] = 1. If the payload size received does not match any value in *Table 706* then the flag RISR[RXSZF] = 1. In both these error conditions the received frame is ignored. The S/W may have to take the necessary steps to recover from such an error.

47.7.6 Frame flow

Following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

47.7.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

47.7.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The

MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO.Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

47.7.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTF] = 1 after each frame transfer.
5. CTS is set depending on the push/pull mode setting defined by MCR[CTSEN].

47.7.6.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[RXEN] and MCR[DRFEN] bits should be set to enable the data frame reception.When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded. In this case, UNSRSR[RXOF] = 1 and an interrupt is generated if the RIER[RXOFIE] = 1.

47.7.6.2 Unsolicited flow

47.7.6.2.1 Unsolicited frame transmit flow

The S/W is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programmed into the Unsolicited Data and Control registers. Once the Payload is programmed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

47.7.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the Mode Configuration Register (MCR) to enable the Tx path.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
   - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
     - UNSTCR[USNDRQ] = 0
     - TISR[TXUNSF] = 1.
   - If UNSTCR[USNDRQ] = 0 then:
     - Program the unsolicited payload in UNSTDR0–UNSTDR8, and can be written up to a payload of frame of a maximum of 288 bits.
     - Program the unsolicited frame header in UNSTCR[UNSHDR].
     - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.
6. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

47.7.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDR0–UNSTDR8.
3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.
3. Then it clears the interrupt (write RISR[RXUNSF] = 1) and also the UNSRSR[URXDV] bit.

47.7.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is defined as a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

47.7.6.3.1 ICLC data transmit flow

Whenever the processor intends to transfer an ICLC frame to the LFAST slave then the following steps are to be followed.

1. Write the ICLC frame payload in the ICR[ICLCPLD] (see *Table 707* for payloads as defined by the standard).
2. Program ICR[SNDICLC] = 1 to initiate the ICLC frame transfer. This bit clears itself when the ICLC frame has been transmitted. The processor needs to poll this bit to make sure it is cleared before setting this bit again (even when ICR[ICLCSEQ] = 1).
3. TISR[TXICLCF] = 1 after the transfer of the ICLC frame.
4. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

To determine whether the ICLC frame has been sent or not, either the ICR[SNDICLC] bit can be polled, or wait for TISR[TXICLCF] = 1.

Table 707. Supported ICLC Payloads

ICLC Code (hex)	ICLC Function
00	Ping Request from LFAST Master to LFAST Slave
01	Reserved
02	Not Supported
04	Not Supported
08	Select TxData Slow (LFAST Slave Rx Interface)
10	Select TxData Fast (LFAST Slave Rx Interface)
20	Select RxData Slow (LFAST Slave Tx Interface)
40	Not Supported
80	Select RxData Fast (LFAST Slave Tx Interface)
31	Enable RxData Interface (LFAST Slave Tx Interface)
32	Disable RxData Interface (LFAST Slave Tx Interface)
34	Clock Test mode Send 101010… continuously using the currently configured clock rate (slow, fast);
FF	Turn payload loopback on
38	Turn Test mode (Loopback and Clock Test) off

Programing model for ICLC frame transmit:

1. Set the ICR[ICLCSEQ] bit (optional).
2. Write the ICLC frame payload in the ICR[ICLCPLD], corresponding to the desired Tx/Rx data rate change, as described in the *Table 707*.
3. Write ICR[SNDICLC] = 1 to initiate the ICLC frame transfer.
4. LFAST master will schedule the ICLC transfer. Data present in ICR[ICLCPLD] is transmitted to the LFAST slave at the old Tx data rate. LFAST slave will configure its Tx/Rx interface data rate accordingly.
5. Reset the ICR[ICLCSEQ] to allow the scheduling of other types of frames, which will be sent at the new data rate.

When ICR[ICLCSEQ] = 0, though the ICLC frame transfer still has highest arbitration priority, other frames may be scheduled if no valid ICLC frame request exists.

Note: ICLC frame transmit will not trigger the internal data rate change of LFAST Master Tx/Rx interface. LFAST master Tx/Rx interface data rate will only be changed, when corresponding SCR[TDR] and SCR[RDR] bits are appropriately configured.

Note: Processor requires to take care that no data frames are to be transmitted while data rate change is happening or ICLC is transmitting frames by setting ICR[ICLCSEQ] bit in Table 707.

47.7.6.3.2 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

The reception of an ICLC frame is indicated by an interrupt to the system. The ICLC status register RISR indicates the type of the ICLC frame received. The Rx block will decode and generate the appropriate signals. An invalid ICLC code besides the ones listed in *Table 707* will cause RISR[RXICF] = 1.

47.7.6.3.2.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF] = 1. The S/W can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame.The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

47.7.6.3.2.2 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

47.7.6.3.2.3 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

47.7.6.3.2.4 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or S/W programs MCR[TXEN] = 1.

47.7.6.3.2.5 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

47.7.6.3.2.6 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

47.7.6.3.2.7 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable .

47.7.6.3.2.8 Ping response ICLC

The LFAST master considers any ICLC frame payload as ping response data. The LFAST masters Rx block will store the received ping data in PISR[RXPNGD] and match it with PICR[PNGPYLD]. RIISR[ICPSF] = 1 if the Ping received matches with the one stored in PICR[PNGPYLD], or RIISR[ICPFF] = 1.

47.7.6.4 CTS flow

47.7.6.4.1 CTS Tx flow

The CTS frame, if enabled by the S/W, indicates the status of the Receive Data FIFO. CTS frame is triggered whenever Rx Data FIFO reaches either the High/Low threshold and no data, ICLC or unsolicited frame is ready for transmission.

Programing model for CTS frame transmit:

- Program RFCR[CTSMX] and RFCR[CTSMN].
- Enable the CTS frame Transmission by programming MCR[CTSEN] = 1.
- Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path.
- The CTS bit sent with a valid frame (for example, data, ICLC and unsolicited) is 1 until the Rx Data FIFO reaches High Threshold.
- When the Rx Data FIFO reaches higher threshold and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 0 and the status bit RISR[RXMXF] = 1.
- When the Rx Data FIFO reaches Low Threshold, due to system side reads and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 1, and status bit RISR[RXMNF] = 1.

47.7.6.4.2 CTS Rx flow

Whenever a CTS frame header or any other frame with valid header is received with CTS bit = 0, then RISR[RXCTSF] = 1. The Tx Interface arbitration for unsolicited frame and data frame transmit request is turned off on reception of frame with CTS bit 0. The arbitration for unsolicited frame and data frame is enabled on reception of frame with CTS bit 1. Frames with LCT type CTS are not stored in the LFAST.

47.7.7 Test and Debug Support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

47.7.7.1 Loopback Test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for S/W should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programmed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 708. Loopback modes

LPMOD[2:0]	Mode Selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by S/W, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

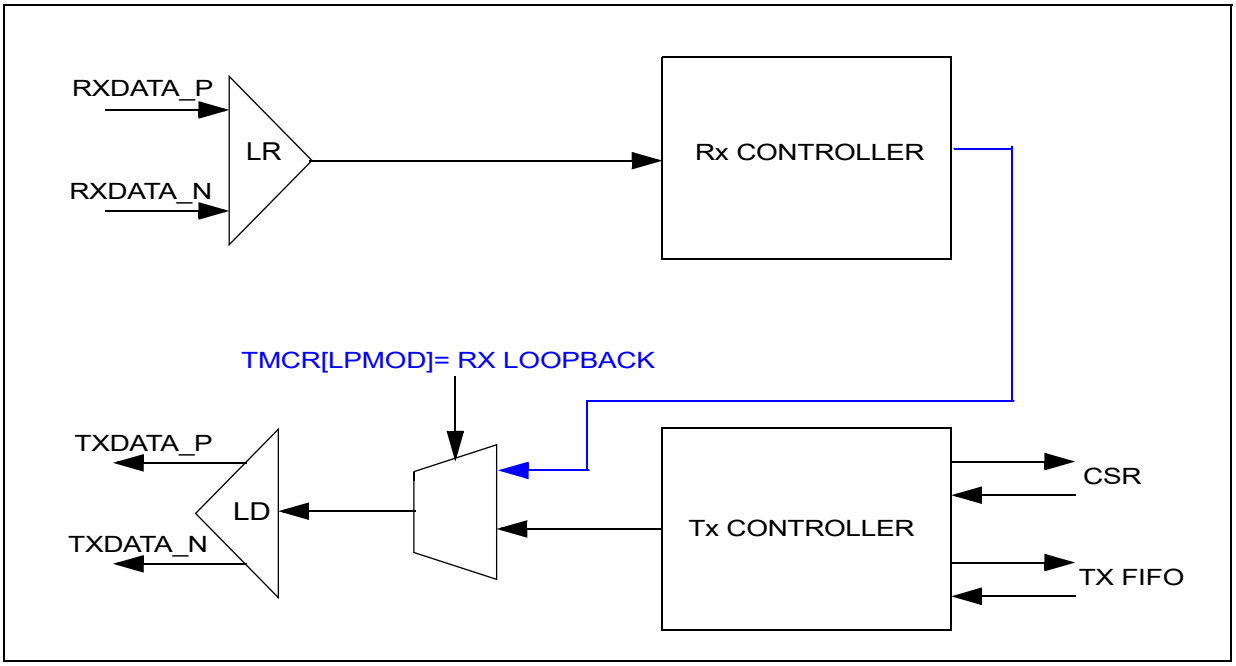
In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.
- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

47.7.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. CTS bit (Bit 0) of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

Figure 688. Rx LoopBack mode



Entry to Rx Loopback mode:

1. S/W programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
   – S/W programs TMCR[LPON] = 1.
   – For Slave Only: - Reception of ICLC Frame with Payload FFh (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

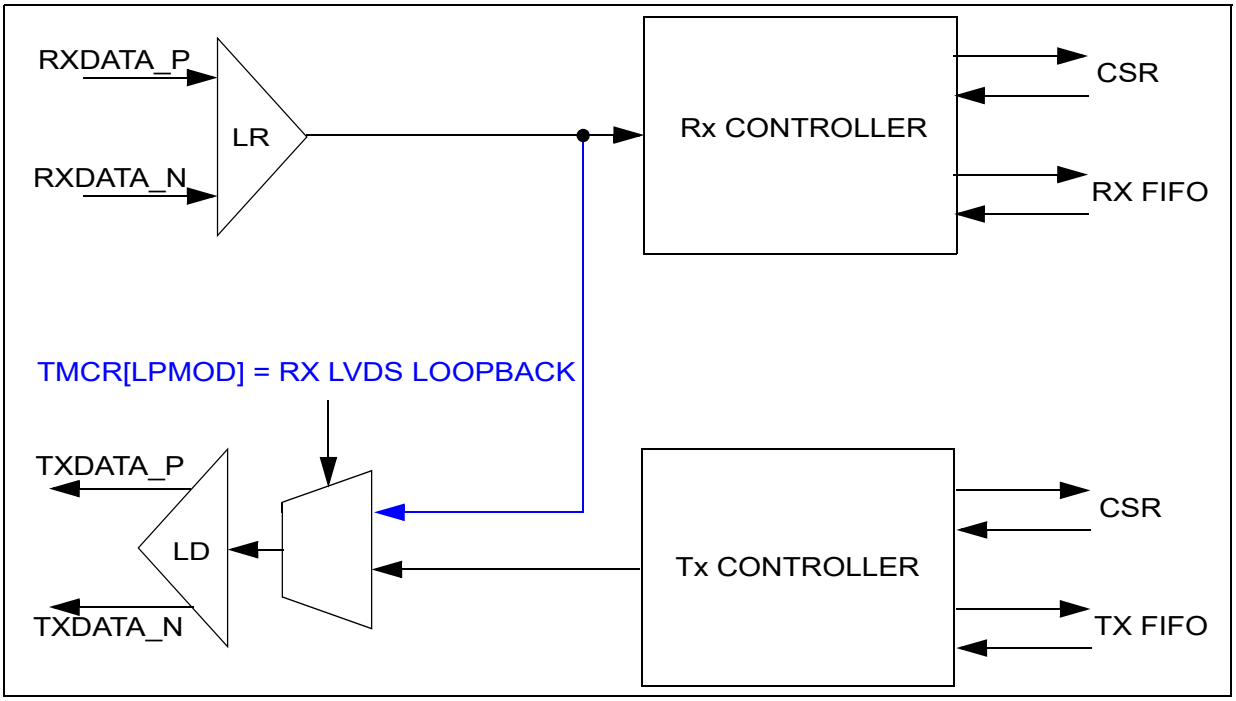
- S/W programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 38h (Test mode off), from LFAST Master

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

47.7.7.1.2 Rx LVDS LoopBack mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via “Rx LVDS LoopBack Mux”. Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

Figure 689. Rx LVDS LoopBack mode



Entry to Rx LVDS Loopback mode:

1. S/W programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:
   – S/W writes TMCR[LPON] = 1.
   – For Slave Only: - Reception of ICLC Frame with Payload = FFh (Loopback mode ON), from LFAST Master.

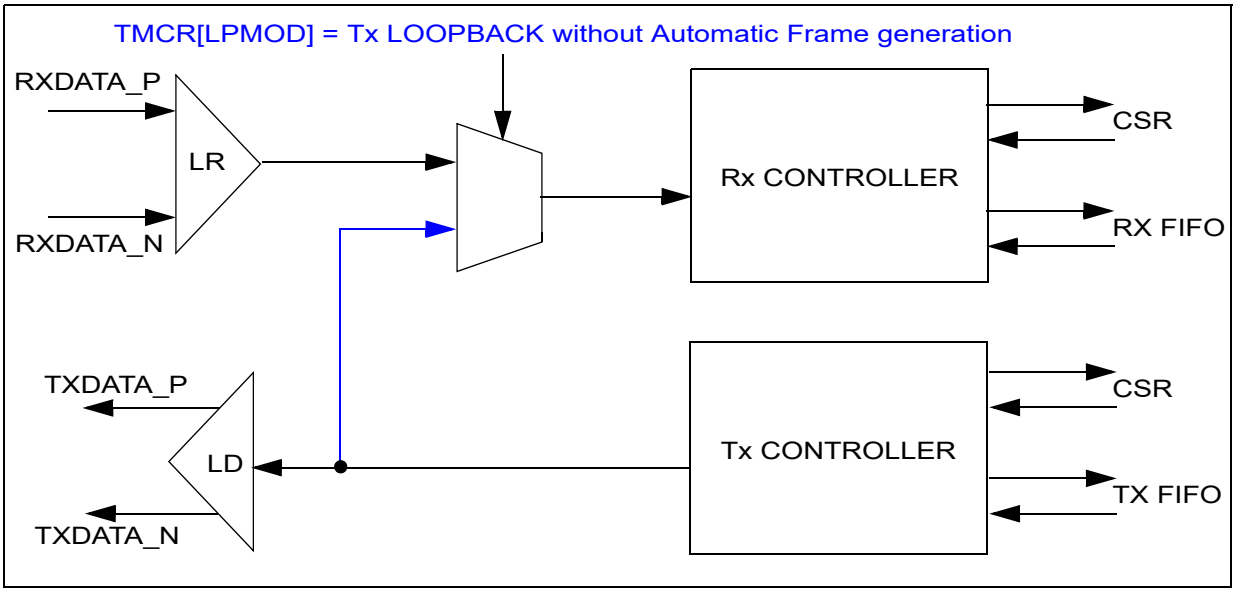
Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave:
  – Transmission of ICLC frame with payload 38h (Test mode off), from LFAST Master.

47.7.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

Figure 690. Tx LoopBack mode without automatic frame generation



Entry to Tx Loopback without Automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
   – S/W programs TMCR[LPON] = 1.
   – For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off), from Tx interface (using unsolicited frame).

All frames and error flags are decoded in this mode.

47.7.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where predefined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal S/W intervention.

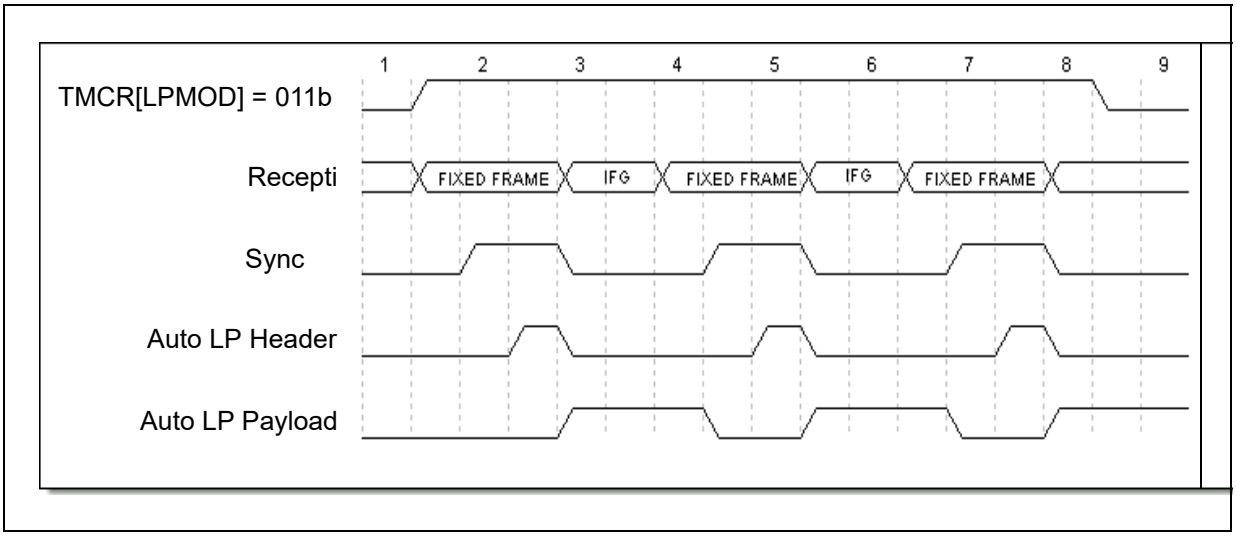
The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted.
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The Status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 13h received.
- GSR[LPCPDV]: Frame with Payload of CBh received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

Figure 691. Automatic Loopback Test status signal timings



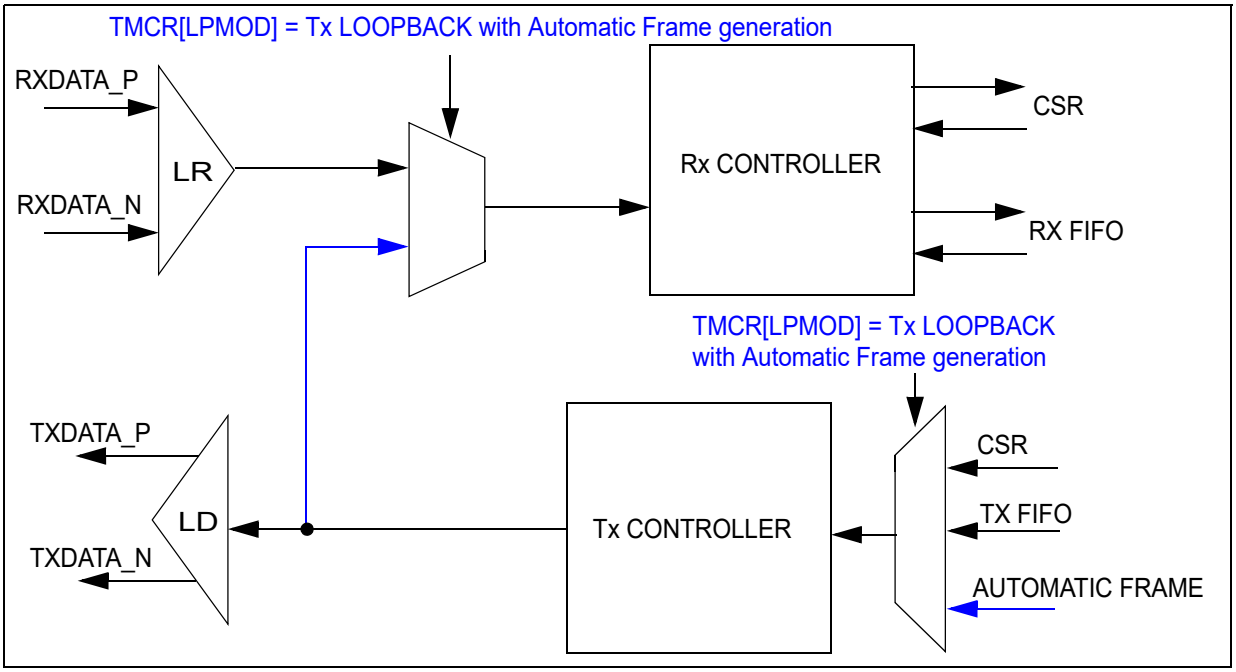
The two Loopback with automatic frame generation modes are:

- Tx Loopback with Automatic frame generation.
- Tx LVDS (external) with Automatic frame generation.

47.7.7.1.4.1 Tx LoopBack mode with automatic frame generation

When TMCR[LPMOD] = 011b (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the S/W. The frames generated have fixed header of 13h and payload of CBh. The successful reception of this frame is indicated by the status signals and registers.

Figure 692. Tx LoopBack mode with automatic frame generation



Entry to Tx Loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
   – S/W programs TMCR[LPON] = 1.
   – For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from the LFAST Master.

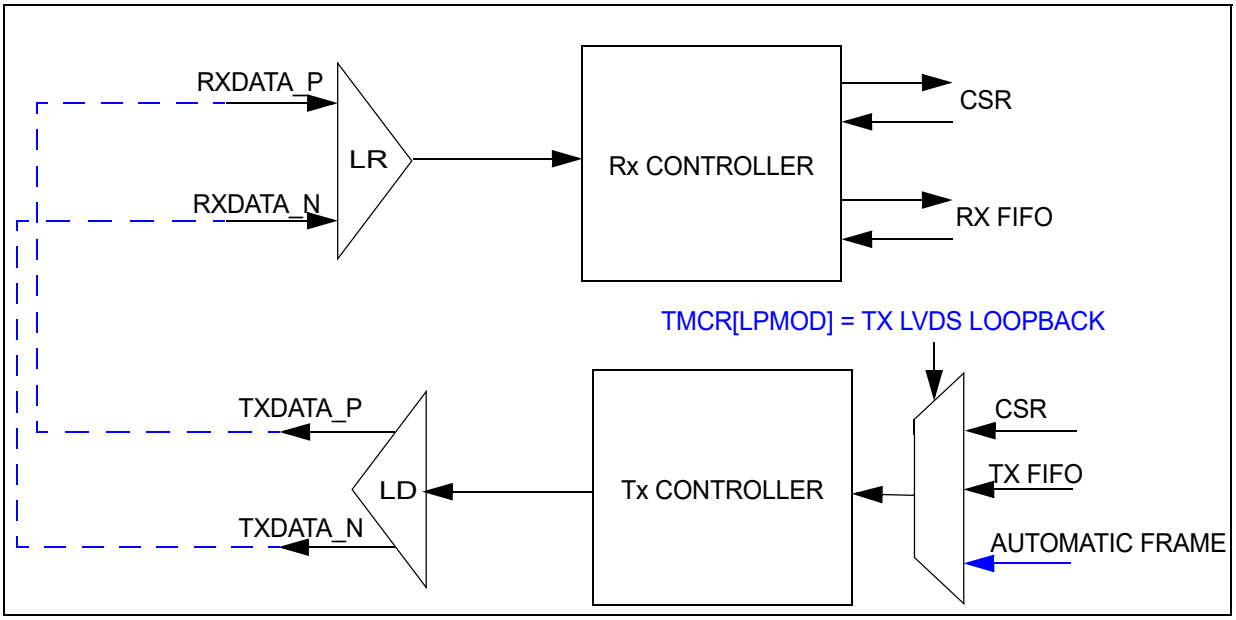
Exit from Tx Loopback with Automatic frame generation mode can be done by the following method:

- S/W programs TMCR[LPON] = 0.

47.7.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

Figure 693. Tx LVDS Loopback (external) mode with automatic frame generation



Entry to Tx LVDS loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
   – S/W programs TMCR[LPON] = 1.
   – For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on) from LFAST Master

Exit from Tx LVDS Loopback with automatic frame generation mode can be done by the following method:

- S/W programs TMCR[LPON] = 0.

47.7.7.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- S/W programs TMCR[CLKTST].
- For Slave Only: Reception of ICLC frame with payload 34h (Clock Test mode on) from LFAST Master.

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be canceled by any of the following methods:

- S/W programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off) from LFAST Master.

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1’s and 0’s.

This mode does not affect the functionality of the Rx Interface.

47.7.8 Interrupts

The LFAST module generates five interrupts to the processor:

- Transmit complete interrupt
- Transmit exception interrupt
- Receive complete interrupt
- Receive exception interrupt
- ICLC receive interrupt

Each interrupt is asserted when any one of their status conditions is met.

47.7.8.1 Transmit complete interrupt

This interrupt indicates a transfer completion of a frame. This interrupt is asserted whenever any of the following bits are set in the TISR and also the corresponding mask bits are set in the TIER.

- TXDTF
- TXICLCF
- TXUNSF
- TXPNGF

Each of these bits can be individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

47.7.8.2 Transmit exception interrupt

This interrupt indicates occurrence of an exception condition in the Tx block. This interrupt is asserted whenever any of the following bits are set in the TISR along with the corresponding mask bits in the TIER.

- TXOVF
- TXIEF

Each of these bits can be individually masked. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

Table 709. Recommended transmit exception handling

Exception	Recommendation
TXOVF	– MCR[TXEN] = 0 – The system level interfaces transmit is reseted and enabled again – For Slave: After reception of ICLC Ping Response Request the MCR[TXEN] may be set – For Master: The MCR[TXEN] should be set and an ICLC frame Ping Response Request should be sent
TXIEF	– The MCR[TXEN] should be set

47.7.8.3 Receive complete interrupt

This interrupt indicates a reception of a frame. This interrupt is asserted whenever any of the following bits is set in the RISR along with the corresponding interrupt enable bit is also set in the RIER.

- RXCTSF
- RXDF
- RXUNSF

Each of these bits are individually maskable. So, the bits that are not required to generate an interrupt to the processor can be masked by clearing its bit in the RIER. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

47.7.8.4 Receive exception interrupt

This interrupt indicates occurrence of an exception condition in Rx block. This interrupt is asserted whenever any of the following bits is set in the RISR and corresponding enable mask bit is also set in the RIER.

- RXLCEF
- RXICF
- RXSZF
- RXUOF
- RXUFF
- RXMXF
- RXMNF
- RXOFF

Each of these bits is individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked by clearing the appropriate bit in the RIER. For details on the cause of assertion of each bit refer to the RISR description. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

Table 710. Recommended receive exception handling mechanism

Exception	Recommendation
RXLCEF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXICF	No action required
RXSZF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXOFF	– The MCR[RXEN] may be cleared – The system level interfaces receive may be reset and enabled, as current transfer may be corrupted
RXUFF	No action required
RXMXF	The system level interfaces receive should be enabled, if not enabled
RXMNF	No action required
RXUNSF	No action required

47.7.8.5 ICLC receive interrupt

This interrupt indicates reception of a valid ICLC frame. This interrupt is asserted whenever any of the following bits is set in the RIISR along with the corresponding enable bit mask bit is set in the RIIER.

- ICPONF
- ICPOFF
- ICTSF
- ICTFF
- ICRSF
- ICRFF
- ICTEF
- ICTDF
- ICCTF
- ICLPF
- ICTOF
- ICPRF
- ICPSF
- ICPFF

Each of these bits is individually maskable. So those bits that are not required to generate an interrupt to the processor can be masked. For details on the cause of assertion of each bit, refer RIISR. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in RIISR.

47.8 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has it own dedicated buffer, and they are not shared between each other.

Table 711. Frames Supported by LFAST interfaces

Frame Type	Tx Buffer (in bits)	Rx Buffer (in bits)	Memory Type
Data Frame	38 × 32 max 6 packets	38 × 32 max 6 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	1 × 7 max 1 packet(1)	1 × 8 max 1 packet(2)	Registers ICR, PISR
CTS Frame	N/A	N/A	N/A

1. Only for MCR[MSEN] = 1.
2. Only for ping response data when MCR[MSEN] = 1.

47.9 Resets

The various blocks in LFAST are reset as described in *Table 712*.

Table 712. Recommended receive exception handling mechanism

Reset	Blocks Reset
Asynchronous Hardware reset	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space
DRFRST bit of Mode Configuration Register (MCR)	Polarity: Active High – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space
DRFEN bit of Mode Configuration Register (MCR)	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Status Registers – SCR[RDR] and SCR[TDR] – TMCR[CLKTST] and TMCR[LPON] – ICR[ICLCSEQ] and ICR[SNDICLC] – PICR[PNGREQ] – UNSTCR[USNDRQ]

47.10 Clocks

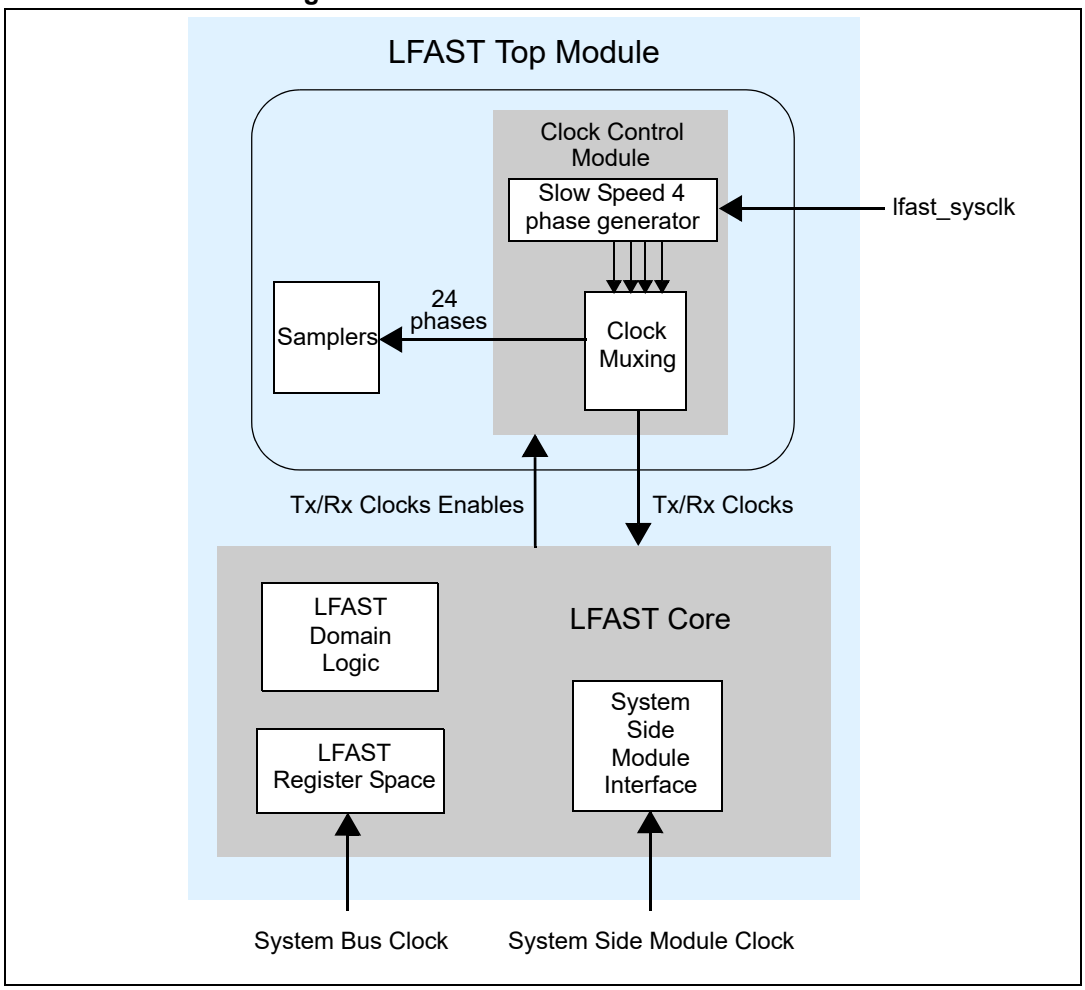
The LFAST mainly works on two clocks:

- System Bus Clock, used to program the registers.
- System Side Module clock, which is synchronous to the System Bus clock.

47.10.1 Clocking strategy

Figure 694 shows the clock domain in which each module functions. The Clock Module will generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

Figure 694. LFAST module clock domains

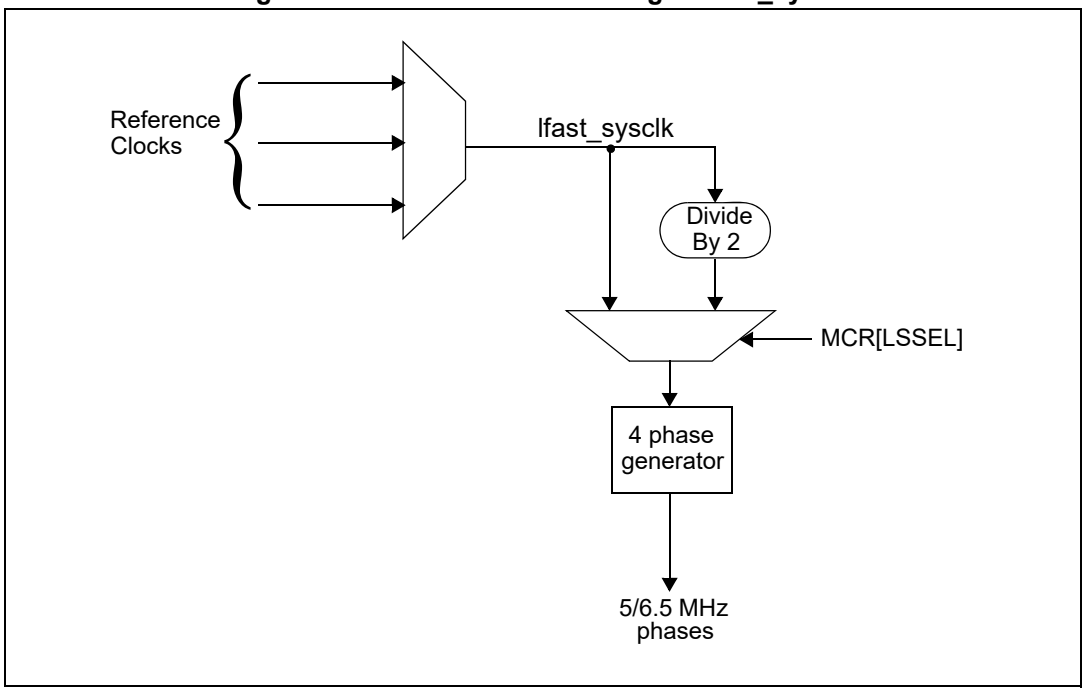


47.10.2 Slow speed clock

47.10.2.1 External muxing

The Clock Control Module will receive lfast_sysclk from the clocking subsystem as shown in *Figure 695*, *Figure 696* and *Figure 697*.

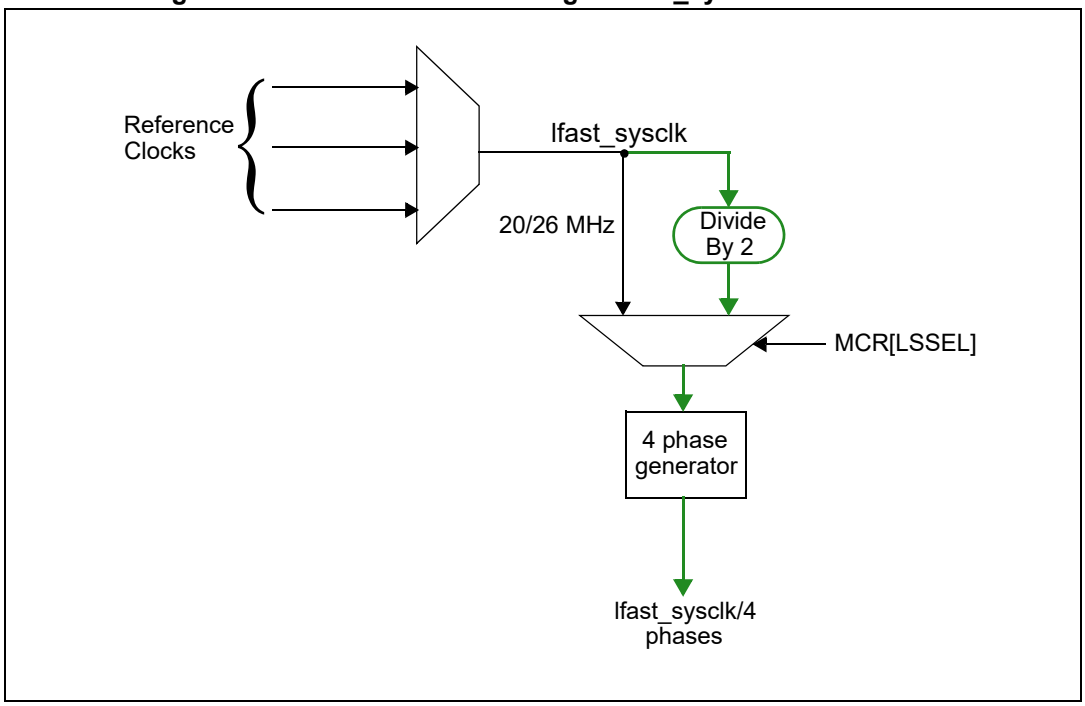
Figure 695. External clock muxing of lfast_sysclk



All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in *Figure 695*.

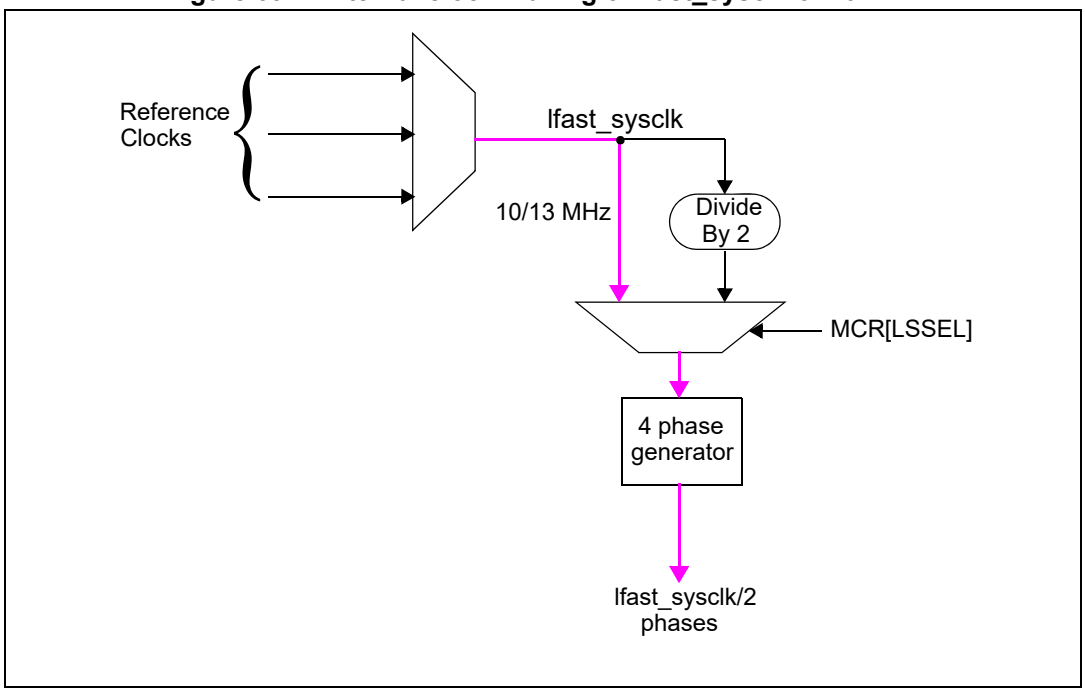
The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in *Figure 696* (selected clock path shown in green).

Figure 696. External clock muxing of lfast_sysclk of 20/26 MHz



In this case, lfast_sysclk is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/2 when MCR[LSSEL] = 0. If lfast_sysclk is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in *Figure 697* (selected clock path selected shown in magenta).

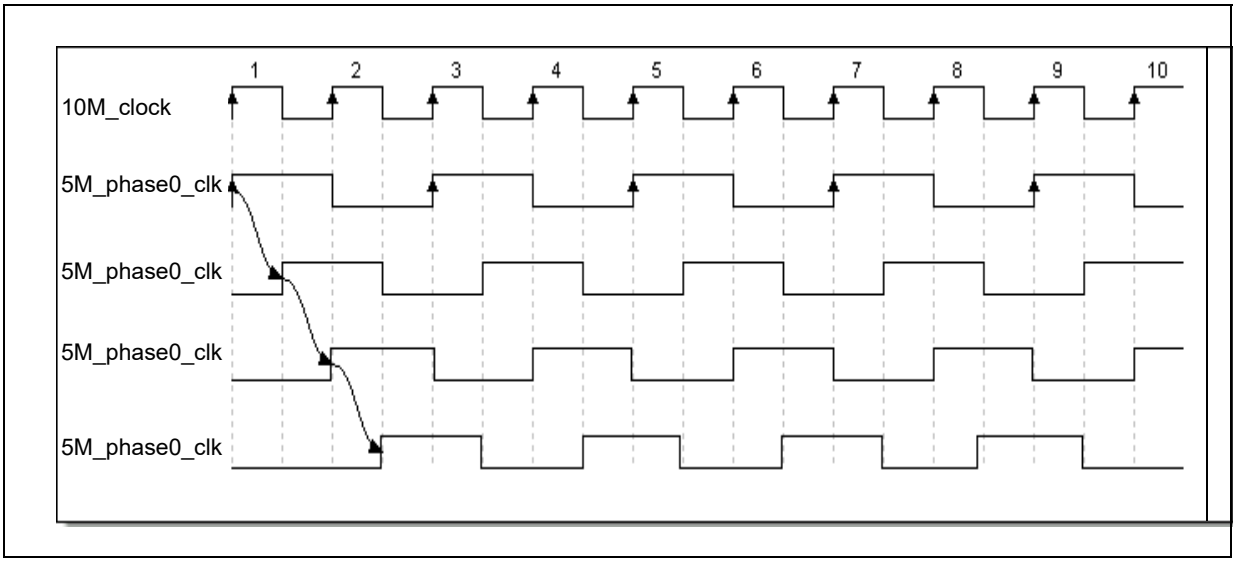
Figure 697. External clock muxing of lfast_sysclk of 10 MHz



47.10.2.2 Slow speed 4-phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. *Figure 698* shows 4 phases getting generated using lfast_sysclk.

Figure 698. Slow speed 4 phases generation

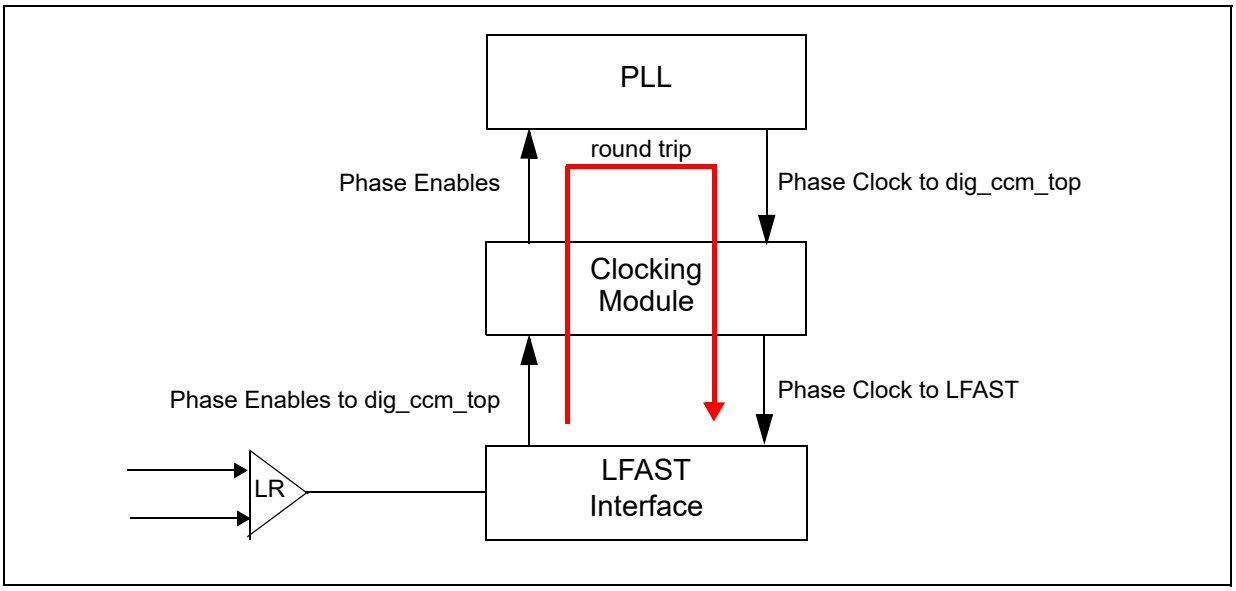


47.10.3 Rx controller clocks

47.10.3.1 Clocking module requirements for high speed phases

The high speed Phases are obtained from the PLL via the Clocking Module as shown in *Figure 699*.

Figure 699. Clock Enables and Clock Paths



The LFAST block will know which of the phases will be required for the different modes of operations. Therefore, the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

47.10.3.2 Clock module requirements for low speed phases

47.10.3.2.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq × 8 MHz clock from the PLL is not needed.

Table 713. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

47.10.4 Tx controller clocks

47.10.4.1 Clocking module requirements for speed phases

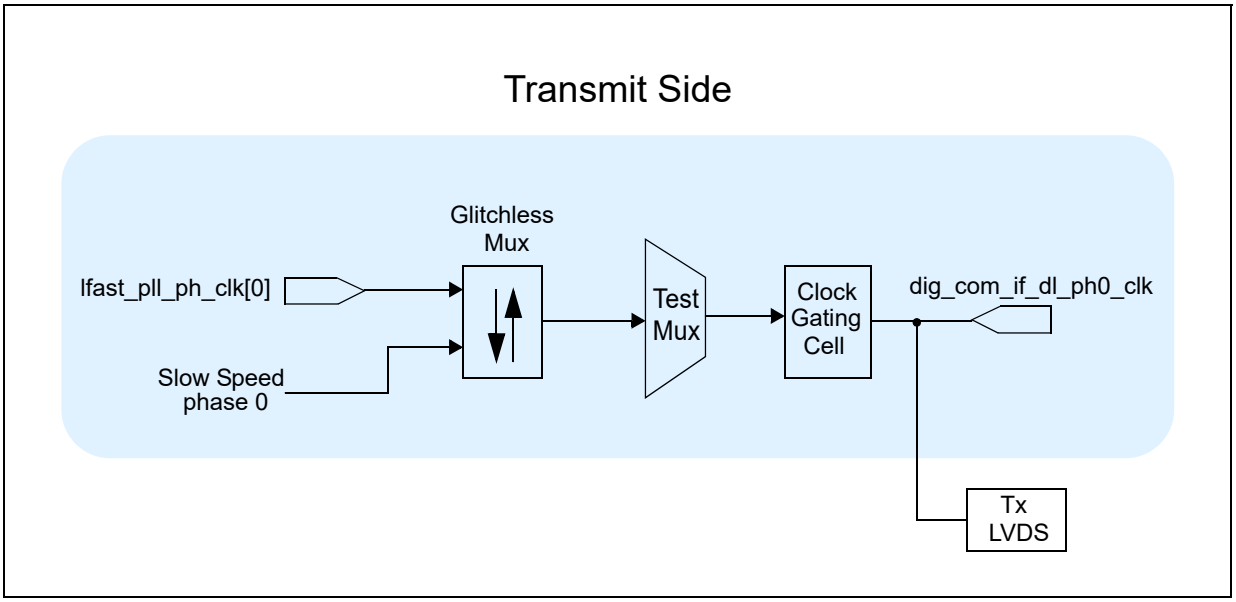
The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See *Table 714*.

Table 714. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

47.10.4.2 Transmit clock muxing

Figure 700. Transmit Clocks Muxing



47.10.4.3 Tx Request Clock Control

The Tx phase 0 clock is enabled when all the resets are negated.

48 Fast Ethernet Controller (FEC)

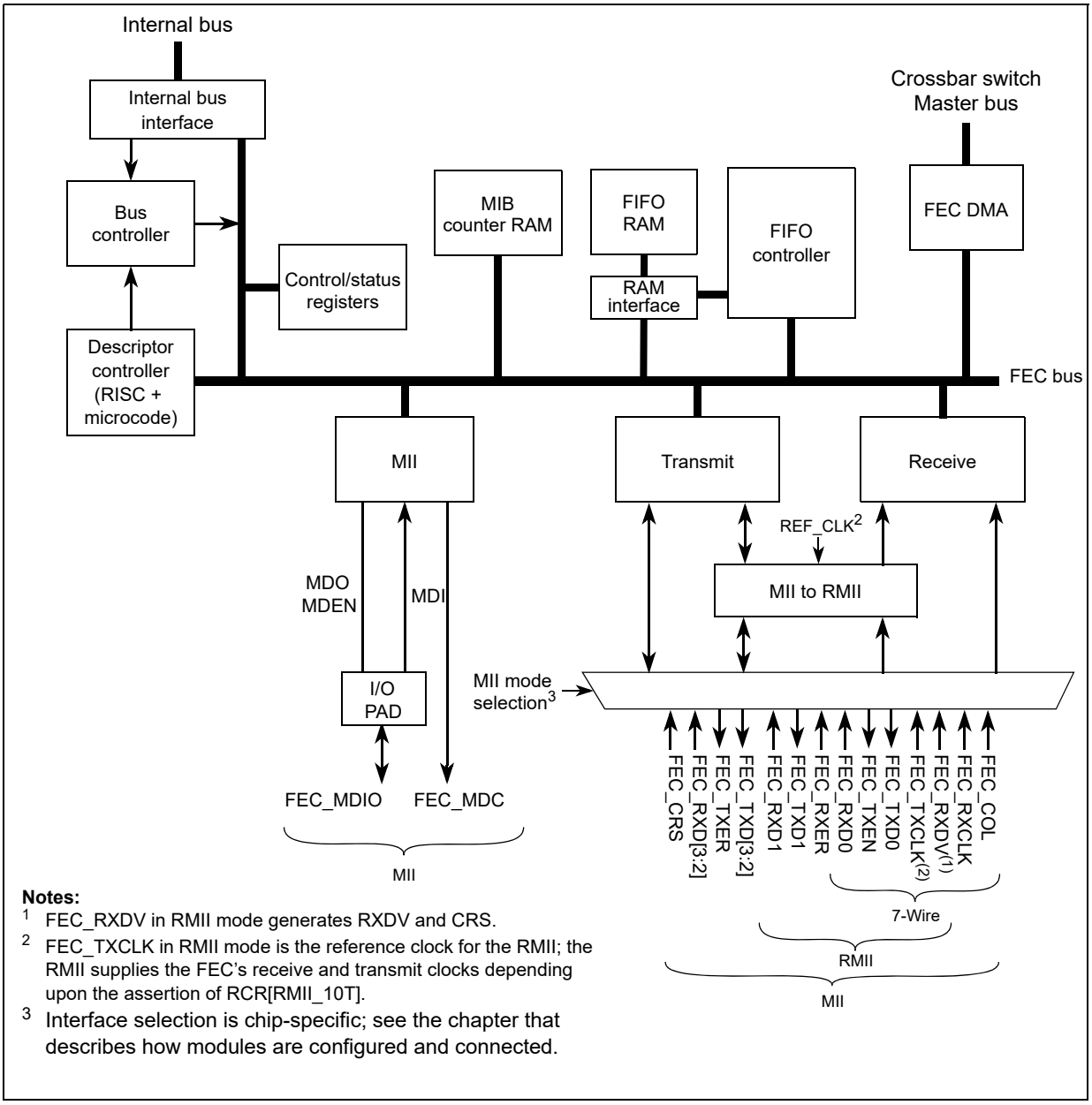
48.1 Introduction

The Fast Ethernet Controller (FEC) is a communication controller that supports 10 and 100 Mbps Ethernet/IEEE 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports five different standard MAC-PHY (physical) interfaces for connection to an external Ethernet transceiver. The FEC supports the 10/100 Mbps MII, 10/100 Mbps reduced MII, and the 10 Mbps-only 7-wire interface.

48.1.1 Block diagram

Figure 701 shows the block diagram of the FEC. The FEC is implemented with a combination of hardware and microcode. The off-chip (Ethernet) interfaces are compliant with industry and IEEE 802.3 standards.

Figure 701. FEC block diagram



Notes:

1 FEC_RXDV in RMII mode generates RXDV and CRS.

2 FEC_TXCLK in RMII mode is the reference clock for the RMII; the RMII supplies the FEC's receive and transmit clocks depending upon the assertion of RCR[RMII_10T].

3 Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

Table 715 provides a brief description of the blocks in *Figure 701*.

Table 715. FEC sub-block descriptions

Block	Description
Control and status registers (CSR)	Provides global control (Ethernet reset and enable) and interrupt managing registers
Descriptor controller	A RISC-based controller providing these functions in the FEC: – Initialization (those internal registers not initialized by you or hardware) – High-level control of the DMA channels (initiating DMA transfers) – Interpreting buffer descriptors – Address recognition for receive frames – Random number generation for transmit collision backoff timer
DMA block	Provides multiple channels allowing transmit data, transmit descriptor, receive data and receive descriptor accesses to all run independently **Note:** In this document, “DMA” refers to the DMA block within the FEC, and is not related to any other DMA controllers that may be present on the chip.
Media independent interface (MII)	Provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of: – The management data clock (FEC_MDC) – The management data input/output lines (FEC_MDIO) of the MII interface
Message information block (MIB)	Maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC, but provides valuable counters for network management. The counters supported are: – The RMON (RFC 1757) Ethernet Statistics group – Some of the IEEE 802.3 counters See the memory map for more information.
RAM	Serves as the focal point of all data flow in the Fast Ethernet controller and divides into transmit and receive FIFOs. The FIFO boundaries are programmable using the FRSR register. User data flows to/from the DMA block from/to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO into the transmit block. Receive data flows from the receive block into the receive FIFO.
Transmit and receive blocks	Provide the Ethernet MAC functionality (with some assist from microcode)

48.1.2 Features

The FEC incorporates the following features:

- Support for the following Ethernet physical interfaces:[p]
  - 100-Mbps IEEE 802.3 MII
  - 10-Mbps IEEE 802.3 MII
  - 100-Mbps reduced media independent interface (RMII)
  - 10-Mbps reduced media independent interface (RMII)
  - 10-Mbps 7-wire interface (industry standard)
- IEEE 802.3 full duplex flow control
- Programmable max frame length supports IEEE 802.1 VLAN tags and priority
- Support for full-duplex operation (200 Mbps throughput) with a minimum internal bus clock rate of 50 MHz
- Support for half-duplex operation (100 Mbps throughput) with a minimum internal bus clock rate of 25 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
  - Frames with broadcast address may be always accepted or always rejected
  - Exact match for single 48-bit individual (unicast) address
  - Hash (64-bit hash) check of individual (unicast) addresses
  - Hash (64-bit hash) check of group (multicast) addresses
  - Promiscuous mode

48.2 Modes of operation

This section describes the FEC operating modes.

48.2.1 Full and half duplex operation

Full duplex mode is for use on point-to-point links between switches or end node to switch. Half duplex mode works in connections between an end node and a repeater or between repeaters. TCR[FDEN] controls duplex mode selection.

When configured for full duplex mode, flow control may be enabled. For more details, see:

- The RFC_PAUSE and TFC_PAUSE fields in the TCR (*Section 48.4.10: Transmit Control Register (TCR)*)
- The RCR[FCE] field (*Section 48.4.9: Receive Control Register (RCR)*)
- *Section 48.5.13: Full duplex flow control*

p. Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

48.2.2 Interface options[q]

The following interface options are supported. A detailed discussion of the interface configurations is provided in *Section 48.5.8: Network interface options*.

48.2.2.1 10 Mbps and 100 Mbps MII interface

The IEEE 802.3 standard defines the media independent interface (MII) for 10/100 Mbps operation. The MAC-PHY interface may be configured to operate in MII mode by programming RCR[MII_MODE] = 1.

The operation speed is determined by the FEC_TXCLK and FEC_RXCLK pins driven by the external transceiver. The transceiver auto-negotiates the speed or software controls it via the serial management interface (FEC_MDC/FEC_MDIO pins) to the transceiver. Refer to the MMFR and MSCR register descriptions, as well as the section on the MII, for a description of how to read and write registers in the transceiver via this interface.

48.2.2.2 10 Mbps and 100 Mbps RMII interface

The reduced media independent interface (RMII) is a low-cost alternative to the IEEE 802.3 MII standard. This interface provides the functionality of the MII interface on a total of 8 pins instead of 18. The RMII interface for 10/100 Ethernet MAC-PHY interface was defined by an industry consortium and is not currently included in the IEEE 802.3 standard. This functionality is selected as described in the chip configuration chapter, and is reflected in the read-only RCR[RMII_MODE] bit. The RCR[RMII_10T] bit determines the speed of operation. The reference clock for RMII is always 50 MHz, but this clock can be divided by 10 within the RCR register to support 10 Mbps operation. The PHY must be configured accordingly.

48.2.2.3 10 Mbps 7-wire interface operation

The FEC supports 7-wire interface used by many 10 Mbps Ethernet transceivers. The RCR[MII_MODE] bit controls this functionality. If this bit is cleared, MII mode is disabled and the 10 Mbps 7-wire mode is enabled.

48.2.3 Address recognition options

The address options supported are:

- Promiscuous
- Broadcast reject
- Individual address (hash or exact match)
- Multicast hash match

Address recognition options are discussed in detail in *Section 48.5.11: Ethernet address recognition*.

48.2.4 Internal loopback

Internal loopback mode is selected via RCR[LOOP]. Loopback mode is discussed in detail in *Section 48.5.16: MII internal and external loopback*.

q. Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

48.3 External signal description

Table 716 describes the various FEC signals, as well as indicates which signals work in available modes.

Table 716. FEC signal descriptions

Signal name	MII	7-wire	RMII	Description(1)
FEC_COL	X	X	—	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full-duplex mode.
FEC_CRS	X	—	—	Carrier sense. When asserted, indicates transmit or receive medium is not idle. In RMII mode, this signal is present on the FEC_RXDV pin.
FEC_MDC	X	—	X	Output clock provides a timing reference to the PHY for data transfers on the FEC_MDIO signal.
FEC_MDIO	X	—	X	Transfers control information between the external PHY and the media-access controller. Data is synchronous to FEC_MDC. This signal is an input after reset. When the FEC operates in 10Mbps 7-wire interface mode, this signal should be connected to VSS.
FEC_RXCLK	X	X	—	Provides a timing reference for FEC_RXDV, FEC_RXD[3:0], and FEC_RXER.
FEC_RXDV	X	X	X	Asserting the FEC_RXDV input indicates PHY has valid nibbles present on the MII. FEC_RXDV must remain asserted from the first recovered nibble of the frame through to the last nibble. Assertion of FEC_RXDV must start no later than the SFD and exclude any EOF. In RMII mode, this pin also generates the CRS signal.
FEC_RXD0	X	X	X	This pin contains the Ethernet input data transferred from PHY to the media-access controller when FEC_RXDV is asserted.
FEC_RXD1	X	—	X	This pin contains the Ethernet input data transferred from PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXD[3:2]	X	—	—	These pins contain the Ethernet input data transferred from PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXER	X	—	X	When asserted with FEC_RXDV, indicates PHY detects an error in the current frame. When FEC_RXDV is not asserted, FEC_RXER has no effect.
FEC_TXCLK	X	X	X	Input clock which provides a timing reference for FEC_TXEN, FEC_TXD[3:0] and FEC_TXER. In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.
FEC_TXD0	X	X	X	The serial output Ethernet data and only valid during the assertion of FEC_TXEN.
FEC_TXD1	X	—	X	This pin contains the serial output Ethernet data and valid only during assertion of FEC_TXEN.
FEC_TXD[3:2]	X	—	—	These pins contain the serial output Ethernet data and valid only during assertion of FEC_TXEN.

Table 716. FEC signal descriptions(Continued)

Signal name	MII	7-wire	RMII	Description[1]
FEC_TXEN	X	X	X	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is negated before the first FEC_TXCLK following the final nibble of the frame.
FEC_TXER	X	—	—	When asserted for one or more clock cycles while FEC_TXEN is also asserted, PHY sends one or more illegal symbols. FEC_TXER has no effect at 10 Mbps or when FEC_TXEN is negated.

1. Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

48.4 Memory map and register definition

The FEC is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs control operation modes and extract global status information. The descriptors pass data buffers and related buffer information between the hardware and software.

Each FEC implementation requires a 1 KB memory map space, which is divided into two sections of 512 bytes each for:

- Control/status registers
- Event/statistic counters held in the MIB block

Table 717 defines the top-level memory map.

Table 717. FEC memory map

Address offset	Function
000h–1FFh	Control/status registers
200h–3FFh	MIB block counters

Table 718 shows the FEC register map. Any address offset not explicitly mentioned in *Table 718* is reserved. Do not modify the content of reserved fields. All accesses to and from the registers must be via 32-bit accesses. There is no support for accesses other than 32-bit. Accessing reserved or unimplemented register spaces in the FEC will not return transfer errors.

The MIB counters (shown in *Table 718*) define the locations in the MIB RAM space where hardware-maintained counters reside. The counters are divided into two groups:

- RMON counters include the Ethernet statistics counters defined in RFC 1757
- Truncated frames counter (only frames with lengths up to 2047 bytes are supported)

The transmit and receive RMON counters are independent, which ensures accurate network statistics when operating in full duplex mode.

The included IEEE counters support the mandatory and recommended counter packages defined in Section 5 of ANSI/IEEE Std. 802.3 (1998 edition). The FEC supports IEEE Basic Package objects, but these do not require counters in the MIB block. In addition, some of the recommended package objects supported do not require MIB counters. Counters for transmit and receive full duplex flow control frames are also included.

Table 718. FEC register map

Address offset	Register	Location
004h	Interrupt Event Register (EIR)	on page 1266
008h	Interrupt Mask Register (EIMR)	on page 1268
010h	Receive Descriptor Active Register (RDAR)	on page 1269
014h	Transmit Descriptor Active Register (TDAR)	on page 1270
024h	Ethernet Control Register (ECR)	on page 1271
040h	MII Management Frame Register (MMFR)	on page 1271
044h	MII Speed Control Register (MSCR)	on page 1273
064h	MIB Control Register (MIBC)	on page 1274
084h	Receive Control Register (RCR)	on page 1274
0C4h	Transmit Control Register (TCR)	on page 1276
0E4h	Physical Address Low Register (PALR)	on page 1277
0E8h	Physical Address High Register and Type Field (PAUR)	on page 1278
0ECh	Opcode/Pause Duration (OPD)	on page 1278
118h	Descriptor Individual Upper Address Register (IAUR)	on page 1279
11Ch	Descriptor Individual Lower Address Register (IALR)	on page 1279
120h	Descriptor Group Upper Address Register (GAUR)	on page 1279
124h	Descriptor Group Lower Address Register (GALR)	on page 1280
144h	Transmit FIFO Watermark (TFWR)	on page 1280
14Ch	FIFO Receive Bound Register (FRBR)	on page 1281
150h	FIFO Receive Start Register (FRSR)	on page 1281
180h	Receive Descriptor Ring Start Register (ERDSR)	on page 1282
184h	Transmit Buffer Descriptor Ring Start Register (ETDSR)	on page 1282
188h	Receive Buffer Size Register (EMRBR)	on page 1283
200h	Count of frames not counted correctly (RMON_T_DROP)	on page 1284
204h	RMON Tx packet count (RMON_T_PACKETS)	on page 1284
208h	RMON Tx broadcast packets (RMON_T_BC_PKT)	on page 1284
20Ch	RMON Tx multicast packets (RMON_T_MC_PKT)	on page 1285
210h	RMON Tx packets with CRC/align error (RMON_T_CRC_ALIGN)	on page 1285
214h	RMON Tx packets < 64 bytes, good CRC (RMON_T_UNDERSIZE)	on page 1285
218h	RMON Tx packets > MAX_FL bytes, good CRC (RMON_T_OVERSIZE)	on page 1286
21Ch	RMON Tx packets < 64 bytes, bad CRC (RMON_T_FRAG)	on page 1286
220h	RMON Tx packets > MAX_FL bytes, bad CRC (RMON_T_JAB)	on page 1286
224h	RMON Tx collision count (RMON_T_COL)	on page 1287
228h	RMON Tx 64 byte packets (RMON_T_P64)	on page 1287
22Ch	RMON Tx 65 to 127 byte packets (RMON_T_P65TO127)	on page 1287

Table 718. FEC register map(Continued)

Address offset	Register	Location
230h	RMON Tx 128 to 255 byte packets (RMON_T_P128TO255)	on page 1288
234h	RMON Tx 256 to 511 byte packets (RMON_T_P256TO511)	on page 1288
238h	RMON Tx 512 to 1023 byte packets (RMON_T_P512TO1023)	on page 1288
23Ch	RMON Tx 1024 to 2047 byte packets (RMON_T_P1024TO2047)	on page 1289
240h	RMON Tx packets with > 2048 bytes (RMON_T_P_GTE2048)	on page 1289
244h	RMON Tx Octets (RMON_T_OCTETS)	on page 1289
248h	Count of transmitted frames not counted correctly (IEEE_T_DROP)	on page 1290
24Ch	Frames transmitted OK (IEEE_T_FRAME_OK)	on page 1290
250h	Frames transmitted with single collision (IEEE_T_1COL)	on page 1290
254h	Frames transmitted with multiple collisions (IEEE_T_MCOL)	on page 1291
258h	Frames transmitted after deferral delay (IEEE_T_DEF)	on page 1291
25Ch	Frames transmitted with late collision (IEEE_T_LCOL)	on page 1291
260h	Frames transmitted with excessive collisions (IEEE_T_EXCOL)	on page 1292
264h	Frames transmitted with Tx FIFO underrun (IEEE_T_MACERR)	on page 1292
268h	Frames transmitted with carrier sense error (IEEE_T_CSERR)	on page 1292
26Ch	Frames transmitted with SQE error (IEEE_T_SQE)	on page 1293
270h	Flow control pause frames transmitted (IEEE_T_FDXFC)	on page 1293
274h	Octet count for frames transmitted without error (IEEE_T_OCTETS_OK)	on page 1293
280h	Count of received frames not counted correctly (RMON_R_DROP)	on page 1294
284h	RMON Rx packet count (RMON_R_PACKETS)	on page 1294
288h	RMON Rx broadcast packets (RMON_R_BC_PKT)	on page 1294
28Ch	RMON Rx multicast packets (RMON_R_MC_PKT)	on page 1295
290h	RMON Rx packets with CRC/Align error (RMON_R_CRC_ALIGN)	on page 1295
294h	RMON Rx packets < 64 bytes, good CRC (RMON_R_UNDERSIZE)	on page 1295
298h	RMON Rx packets > MAX_FL bytes, good CRC (RMON_R_OVERSIZE)	on page 1296
29Ch	RMON Rx packets < 64 bytes, bad CRC (RMON_R_FRAG)	on page 1296
2A0h	RMON Rx packets > MAX_FL bytes, bad CRC (RMON_R_JAB)	on page 1296
2A4h	Reserved (RMON_R_RESVD_0)	on page 1297
2A8h	RMON Rx 64 byte packets (RMON_R_P64)	on page 1297
2ACh	RMON Rx 65 to 127 byte packets (RMON_R_P65TO127)	on page 1297
2B0h	RMON Rx 128 to 255 byte packets (RMON_R_P128TO255)	on page 1298
2B4h	RMON Rx 256 to 511 byte packets (RMON_R_P256TO511)	on page 1298
2B8h	RMON Rx 512 to 1023 byte packets (RMON_R_P512TO1023)	on page 1298
2BCh	RMON Rx 1024 to 2047 byte packets (RMON_R_P1024TO2047)	on page 1299
2C0h	RMON Rx packets with > 2048 bytes (RMON_R_P_GTE2048)	on page 1299

Table 718. FEC register map(Continued)

Address offset	Register	Location
2C4h	RMON Rx octets (RMON_R_OCTETS)	*on page 1299*
2C8h	Count of received frames not counted correctly (IEEE_R_DROP)	*on page 1300*
2CCh	Frames received OK (IEEE_R_FRAME_OK)	*on page 1300*
2D0h	Frames received with CRC error (IEEE_R_CRC)	*on page 1300*
2D4h	Frames received with alignment error (IEEE_R_ALIGN)	*on page 1301*
2D8h	Receive FIFO overflow count (IEEE_R_MACERR)	*on page 1301*
2DCh	Flow control pause frames received (IEEE_R_FDXFC)	*on page 1301*
2E0h	Octet count for frames received without error (IEEE_R_OCTETS_OK)	*on page 1302*

48.4.1 Interrupt Event Register (EIR)

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts which may occur in normal operation are GRA, TXF, TXB, RXF, RXB, and MII. Interrupts resulting from errors/problems detected in the network or transceiver are HBERR, BABR, BABT, LC, and RL. Interrupts resulting from internal errors are HBERR and UN.

Some of the error interrupts are independently counted in the MIB block counters:

- HBERR - IEEE_T_SQE
- BABR - RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
- BABT - RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)
- LATE_COL - IEEE_T_LCOL
- COL_RETRY_LIM - IEEE_T_EXCOL
- XFIFO_UN - IEEE_T_MACERR

Software may choose to mask off these interrupts because these errors are visible to network management via the MIB counters.

Offset: 004h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	0	0	0
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

	16	17	18	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 702. Interrupt Event Register (EIR)

Table 719. EIR field descriptions

Field	Description
HBERR	Heartbeat error. Indicates TCR[HBC] is set and that the COL input was not asserted within the heartbeat window following a transmission.
BABR	Babbling receive error. Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
BABT	Babbling transmit error. Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused by a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
GRA	Graceful stop complete. Indicates the graceful stop is complete. During graceful stop the transmitter is placed into a pause state after completion of the frame currently being transmitted. This bit is set by one of these conditions: – A graceful stop initiated by the setting of the TCR[GTS] bit is now complete. – A graceful stop initiated by the setting of the TCR[TFC_PAUSE] bit is now complete. – A graceful stop initiated by the reception of a valid full duplex flow control pause frame is now complete. See *Section 48.5.13: Full duplex flow control*.
TXF	Transmit frame interrupt. Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
TXB	Transmit buffer interrupt. Indicates a transmit buffer descriptor has been updated.
RXF	Receive frame interrupt. Indicates a frame has been received and the last corresponding buffer descriptor has been updated.
RXB	Receive buffer interrupt. Indicates a receive buffer descriptor not the last in the frame has been updated.
MII	MII interrupt. Indicates the MII has completed the data transfer requested.
EBERR	Ethernet bus error. Indicates a system bus error occurred when a DMA transaction is underway. When the EBERR bit is set, ECR[ETHER_EN] is cleared, halting frame processing by the FEC. When this occurs, software needs to ensure that the FIFO controller and DMA also soft reset.
LC	Late collision. Indicates a collision occurred beyond the collision window (slot time) in half duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.

Table 719. EIR field descriptions(Continued)

Field	Description
RL	Collision retry limit. Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half duplex mode.
UN	Transmit FIFO underrun. Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.

48.4.2 Interrupt Mask Register (EIMR)

The EIMR controls which interrupt events are allowed to generate actual interrupts. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR bit (write 1 to clear) or a 0 is written to the EIMR bit.

Each bit in this register corresponds to an interrupt source defined by the EIR. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. At every processor clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR bit reflects the state of the interrupt signal even if the corresponding EIMR bit is set.

Offset: 008h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	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 703. Ethernet Interrupt Mask Register (EIMR)

Table 720. EIMR field descriptions

Field	Description
HBERR	Heartbeat error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
BABR	Babbling receive error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
BABT	Babbling transmit error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.

Table 720. EIMR field descriptions(Continued)

Field	Description
GRA	Graceful stop complete interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
TXF	Transmit frame interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
TXB	Transmit buffer interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
RXF	Receive frame interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
RXB	Receive buffer interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
MII	MII interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
EBERR	Ethernet bus error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
LC	Late collision interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
RL	Collision retry limit interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
UN	Transmit FIFO underrun interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.

48.4.3 Receive Descriptor Active Register (RDAR)

RDAR is a command register, written by the user, indicating the receive descriptor ring is updated (the driver produced empty receive buffers with the empty bit set).

When the register is written, the RDAR bit is set. This is independent of the data actually written by the user. When set, the FEC polls the receive descriptor ring and processes receive frames (if ECR[ETHER_EN] is also set). After the FEC polls a receive descriptor whose empty bit is not set, FEC clears the RDAR bit and ceases receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors are placed into the receive descriptor ring.

The RDAR is cleared at reset and when ECR[ETHER_EN] is cleared.

Offset: 010h 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	RDAR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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 704. Receive Descriptor Active Register (RDAR)

Table 721. RDAR field descriptions

Field	Description
RDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC when no additional empty descriptors remain in the receive ring. Also cleared when ECR[ETHER_EN] is cleared.

48.4.4 Transmit Descriptor Active Register (TDAR)

The TDAR is a command register which the user writes to indicate the transmit descriptor ring is updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor).

When the register is written, the TDAR field is set. This value is independent of the data actually written by the user. When set, the FEC polls the transmit descriptor ring and processes transmit frames (provided ECR[ETHER_EN] is also set). After the FEC polls a transmit descriptor that is a ready bit not set, FEC clears the TDAR field and ceases transmit descriptor ring polling until the user sets the bit again, signifying additional descriptors are placed into the transmit descriptor ring.

The TDAR field is cleared when any of the following events occur:

- Chip reset
- When ECR[ETHER_EN] is cleared
- When ECR[RESET] is set

Offset: 014h 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	TDAR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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 705. Transmit Descriptor Active Register (TDAR)

Table 722. TDAR field descriptions

Field	Description
TDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHER_EN] is cleared.

48.4.5 Ethernet Control Register (ECR)

ECR is a read/write user register, though hardware may alter fields in this register as well. The ECR enables/disables the FEC.

Offset: 024h Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ETHER_EN	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 706. Ethernet Control Register (ECR)

Table 723. ECR field descriptions

Field	Description
ETHER_EN	When this bit is set, FEC is enabled, and reception and transmission are possible. When this bit is cleared, reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. The buffer descriptor(s) for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is cleared, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware alters the ETHER_EN bit under the following conditions: – ECR[RESET] is set by software, in which case ETHER_EN is cleared. – An error condition causes the EIR[EBERR] bit to set, in which case ETHER_EN is cleared.
RESET	When this bit is set, the equivalent of a hardware reset is performed but it is local to the FEC. ECR[ETHER_EN] is cleared and all other FEC registers take their reset values. Also, any transmission/reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately eight internal bus clock cycles after this bit is set.

48.4.6 MII Management Frame Register (MMFR)

The MMFR is used to communicate with the attached MII compatible PHY device(s), providing read/write access to their MII registers. Performing a write to the MMFR causes a management frame to be sourced unless the MSCR is programmed to 0. If MSCR is cleared while MMFR is written and then MSCR is written with a non-zero value, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

The MMFR does not reset to any predefined value.

Offset: 040h Access: User read/write

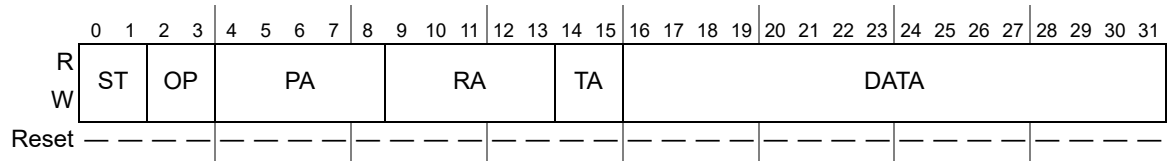


Figure 707. MII Management Frame Register (MMFR)

Table 724. MMFR field descriptions

Field	Description
ST	Start of frame delimiter. These bits must be programmed to 0b01 for a valid MII management frame.
OP	Operation code. 00 Write frame operation, but not MII compliant. 01 Write frame operation for a valid MII management frame. 10 Read frame operation for a valid MII management frame. 11 Read frame operation, but not MII compliant.
PA	PHY address. This field specifies one of up to 32 attached PHY devices.
RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
TA	Turn around. This field must be programmed to 10 to generate a valid MII management frame.
DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

To perform a read or write operation on the MII Management Interface, write the MMFR. To generate a valid read or write management frame, ST field must be written with a 01 pattern, and the TA field must be written with a 10. If other patterns are written to these fields, a frame is generated, but does not comply with the IEEE 802.3 MII definition.

To generate an IEEE 802.3-compliant MII Management Interface write frame (write to a PHY register), write {01 01 PHYAD REGAD 10 DATA} to the MMFR. Writing this pattern causes the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time, contents of the MMFR are altered as the contents are serially shifted and are unpredictable if read by the user. After the write management frame operation completes, the MII interrupt is generated. At this time, contents of the MMFR match the original value written.

To generate an MII management interface read frame (read a PHY register), write {01 10 PHYAD REGAD 10 XXXX} to the MMFR (the content of the DATA field is unimportant). Writing this pattern causes the control logic to shift out the data in the MMFR register following a preamble generated by the control state machine. During this time, contents of the MMFR register are altered as the contents are serially shifted and are unpredictable if read by the user. After the read management frame operation completes, the MII interrupt is generated. At this time, the contents of the MMFR register match the original value written except for the DATA field whose contents are replaced by the value read from the PHY register.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the MII interrupt to avoid writing to the MMFR while frame generation is in progress.

48.4.7 MII Speed Control Register (MSCR)

The MSCR:

- Provides control of the MII clock (FEC_MDC pin) frequency
- Allows a preamble drop on the MII management frame

Offset: 044h 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	DIS_PRE	MII_SPEED						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 708. MII Speed Control Register (MSCR)

Table 725. MSCR field descriptions

Field	Description
DIS_PRE	Setting this bit causes the preamble (32 ones) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
MII_SPEED	Controls the frequency of the MII management interface clock (FEC_MDC) relative to the internal bus clock. A value of 0 in this field turns off the FEC_MDC and leaves it in low voltage state. Any non-zero value results in the FEC_MDC frequency of 1/(MII_SPEED × 2) of the internal bus frequency.

The MII_SPEED field must be programmed with a value to provide an FEC_MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be set to 0 to turn off the FEC_MDC. The FEC_MDC generated has a 50% duty cycle except when MII_SPEED changes during operation (change takes effect following a rising or falling edge of FEC_MDC).

If the internal bus clock is 25 MHz, programming this register to 0000_0005h results in an FEC_MDC as stated in the equation below.

Equation 31

$$25\ \text{MHz} \times \frac{1}{5 \times 2} = 2.5\ \text{MHz}$$

Table 726 shows optimum values for MII_SPEED as a function of internal bus clock frequency.

Table 726. Programming examples for MSCR

Internal FEC clock frequency	MSCR[MII_SPEED]	FEC_MDC frequency
25 MHz	5h	2.50 MHz
33 MHz	7h	2.36 MHz

Table 726. Programming examples for MSCR(Continued)

Internal FEC clock frequency	MSCR[MII_SPEED]	FEC_MDC frequency
40 MHz	8h	2.50 MHz
50 MHz	Ah	2.50 MHz
66 MHz	Eh	2.36 MHz
80 MHz	10h	2.50 MHz
83 MHz	10h	2.59 MHz

48.4.8 MIB Control Register (MIBC)

The MIBC is a read/write register controlling and observing the state of the MIB block. User software accesses this register if there is a need to disable the MIB block operation. For example, to clear all MIB counters in RAM:

1. Disable the MIB block
2. Clear all the MIB RAM locations
3. Enable the MIB block

The MIB_DIS bit is reset to 1. See the memory map for the locations of the MIB counters.

Offset: 064h 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	MIB_DIS	MIB_IDLE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	1	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

Figure 709. MIB Control Register (MIBC)

Table 727. MIBC field descriptions

Field	Description
MIB_DIS	If MIB_DIS = 1, the MIB logic halts and not update any MIB counters.
MIB_IDLE	If MIB_IDLE = 1, the MIB block is not currently updating any MIB counters.

48.4.9 Receive Control Register (RCR)

RCR controls the operational mode of the receive block and must be written only when ECR[ETHER_EN] is cleared (initialization time).

Offset: 084h 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	MAX_FL										
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	0	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	RMII_ECHO	RMII_LOOP	RMII_10T	RMII_MODE(1)	0	0	FCE	BC_REJ	PROM	MII_MODE(1)	DRT	LOOP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 710. Receive Control Register (RCR)

1. Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

Table 728. RCR field descriptions

Field	Description
MAX_FL	Maximum frame length. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes the BABT interrupt to occur. Receive frames longer than MAX_FL causes the BABR interrupt to occur and sets the LG bit in the end of frame receive buffer descriptor. The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
RMII_ECHO	RMII Echo. Enables RMII echo mode. RMII 2-bit receive data is processed through the RMII receive logic to the FEC and also routes through the RMII's transmit logic to become RMII 2-bit transmit data out. 0 Normal operation. 1 RMII echo mode enabled. **Note:** When RMII_ECHO is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_LOOP = 0 and LOOP = 0.
RMII_LOOP	RMII loopback. Enables RMII loopback mode. Causes the MII transmit outputs from the Ethernet controller to loop back to the Ethernet controllers's MII receive inputs through the RMII transmit/receive logic. 0 Normal operation. 1RMII loopback mode enabled. **Note:** When RMII_LOOP is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_ECHO = 0, LOOP = 0, and TCR*n*[FDEN] = 1.
RMII_10T	RMII 10-Base T. Enables 10Mbps mode of the RMII. Determines the clock frequency of the clock source to the FEC logic to support 10/100Mbps operations. 0 100 Mbps operation. The 50 MHz RMII reference clock on FEC_TXCLK is sent to the RMII, while a divided-by-2 version (25 MHz) is sent to the FEC. 1 10 Mbps operation. The 50 MHz RMII reference clock on FEC_TXCLK is divided by 10 (5 MHz) and sent to the RMII, while a divided-by-20 version (2.5 MHz) is sent to the FEC.

Table 728. RCR field descriptions(Continued)

Field	Description
RMII_MODE	RMII Mode. Indicates if the FEC is in RMII or MII/7-wire mode. See the chip configuration chapter for more details. 0 FEC is in true MII mode. 1 FEC is in RMII mode. **Note:** Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.
FCE	Flow control enable. If asserted, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration.
BC_REJ	Broadcast frame reject. If asserted, frames with DA (destination address) equal to FFFF_FFFF_FFFF are rejected unless the PROM bit is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the M (MISS) is set in the receive buffer descriptor.
PROM	Promiscuous mode. All frames are accepted regardless of address matching.
MII_MODE	Media independent interface mode. Selects the external interface mode for transmit and receive blocks. 07-wire mode (used only for serial 10 Mbps) 1MII or RMII mode as indicated by the RMII_MODE bit **Note:** Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.
DRT	Disable receive on transmit. 0 Receive path operates independently of transmit (use for full duplex or to monitor transmit activity in half duplex mode). 1 Disable reception of frames while transmitting (normally used for half duplex mode).
LOOP	Internal loopback. If set, transmitted frames are looped back internal to the device and transmit output signals are not asserted. The internal bus clock substitutes for the FEC_TXCLK when LOOP is asserted. DRT must be set to 0 when setting LOOP.

48.4.10 Transmit Control Register (TCR)

The TCR configures the transmit block.

Offset: 0C4h 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	RFC_PAUSE	TFC_PAUSE	FDEN	HBC	GTS
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 711. Transmit Control Register (TCR)

Table 729. TCR field descriptions

Field	Description
RFC_PAUSE	Receive frame control pause. This read-only status bit is asserted when a full duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete.
TFC_PAUSE	Transmit frame control pause. Transmits a PAUSE frame when asserted. When this bit is set, the MAC stops transmission of data frames after the current transmission is complete. At this time, GRA interrupt in the EIR is asserted. With transmission of data frames stopped, MAC transmits a MAC Control PAUSE frame. Next, the MAC clears the TFC_PAUSE bit and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC Control PAUSE frame.
FDEN	Full duplex enable. If set, frames transmit independent of carrier sense and collision inputs. This field must be modified only when ECR[ETHER_EN] = 0.
HBC	Heartbeat control. If set, the heartbeat check performs following end of transmission and the HB bit in the status register is set if the collision input does not assert within the heartbeat window. This field must be modified only when ECR[ETHER_EN] = 0.
GTS	Graceful transmit stop. When this bit is set, MAC stops transmission after any frame currently transmitted is complete and GRA interrupt in the EIR is asserted. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this situation, program ECR[ETHER_EN] = 0 following the GRA interrupt.

48.4.11 Physical Address Lower Register (PALR)

PALR contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the 6-byte source address field when transmitting PAUSE frames.

This register is not reset and you must initialize it.

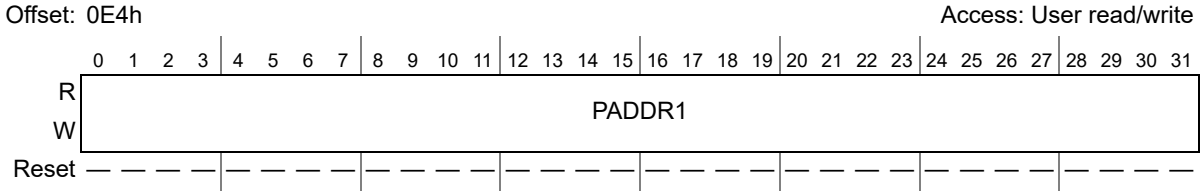


Figure 712. Physical Address Lower Register (PALR)

Table 730. PALR field descriptions

Field	Description
PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

48.4.12 Physical Address Upper Register (PAUR)

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte Source Address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (8808h) for transmission of PAUSE frames. The upper 16 bits of this register are not reset and you must initialize them.

Offset: 0E8h 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	PADDR2																TYPE															
W																																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

Figure 713. Physical Address Upper Register (PAUR)

Table 731. PAUR field descriptions

Field	Description
PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
TYPE	Type field in PAUSE frames. These 16 read-only bits are a constant value of 8808h.

48.4.13 Opcode/Pause Duration Register (OPD)

The OPD contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0001h. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field.

The lower 16 bits of this register are not reset and you must initialize them.

Offset: 0ECh 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	OPCODE																PAUSE_DUR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 714. Opcode/Pause Duration Register (OPD)

Table 732. OPD field descriptions

Field	Description
OPCODE	Opcode field used in PAUSE frames. These read-only bits are a constant, 0001h.
PAUSE_DUR	Pause Duration field used in PAUSE frames.

48.4.14 Descriptor Individual Upper Address Register (IAUR)

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA.

This register is not reset and you must initialize it.

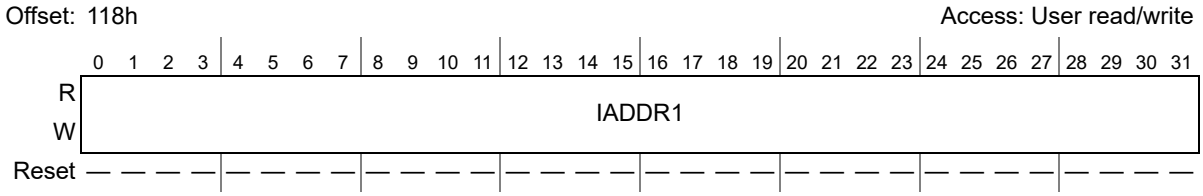


Figure 715. Descriptor Individual Upper Address Register (IAUR)

Table 733. IAUR field descriptions

Field	Description
IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

48.4.15 Descriptor Individual Lower Address Register (IALR)

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA.

This register is not reset and you must initialize it.

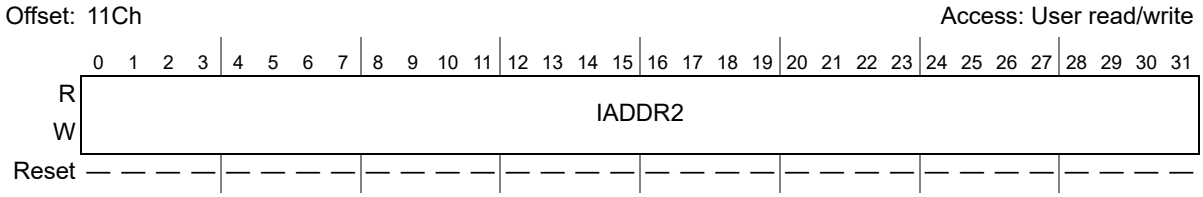


Figure 716. Descriptor Individual Lower Address Register (IALR)

Table 734. IALR field descriptions

Field	Description
IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

48.4.16 Descriptor Group Upper Address Register (GAUR)

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address.

This register is not reset and you must initialize it.

Offset: 120h Access: User read/write

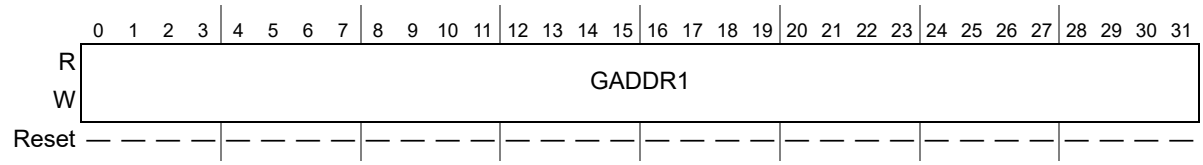


Figure 717. Descriptor Group Upper Address Register (GAUR)

Table 735. GAUR field descriptions

Field	Description
GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

48.4.17 Descriptor Group Lower Address Register (GALR)

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address.

This register is not reset and you must initialize it.

Offset: 124h Access: User read/write

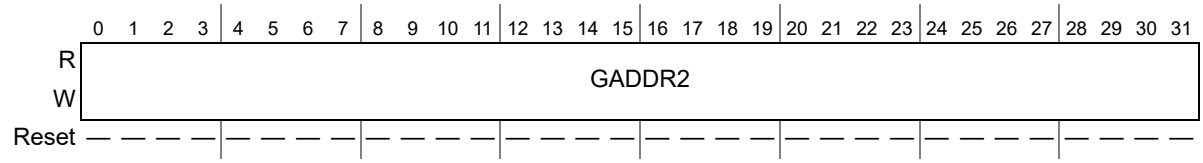


Figure 718. Descriptor Group Lower Address Register (GALR)

Table 736. GALR field descriptions

Field	Description
GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

48.4.18 Transmit FIFO Watermark Register (TFWR)

The TFWR controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement (worst case bus access latency by the transmit data DMA channel).

Offset: 144h 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	0	0	TFWR	
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 719. Transmit FIFO Watermark Register (TFWR)

Table 737. TFWR field descriptions

Field	Description
TFWR	Number of bytes written to transmit FIFO before transmission of a frame begins 00 64 bytes written 01 64 bytes written 10 128 bytes written 11 192 bytes written

48.4.19 FIFO Receive Bound Register (FRBR)

FRBR indicates the upper address bound of the FIFO RAM. Drivers can use this value, along with the FRSR, to appropriately divide the available FIFO RAM between the transmit and receive data paths.

Offset: 14Ch Access: User 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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	R_BOUND								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	1	1	0	0	0	0	0	0	0	0	0

Figure 720. FIFO Receive Bound Register (FRBR)

Table 738. FRBR field descriptions

Field	Description
R_BOUND	Highest valid FIFO RAM address.

48.4.20 FIFO Receive Start Register (FRSR)

FRSR indicates the starting address of the receive FIFO. FRSR marks the boundary between the transmit and receive FIFOs. The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FRSR. The receive FIFO uses addresses from FRSR to FRBR inclusive.

Hardware initializes the FRSR at reset. FRSR only needs to be written to change the default value.

Offset: 150h 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	1	R_FSTART								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	1	0	1	0	0	0	0	0	0	0	0

Figure 721. FIFO Receive Start Register (FRSR)

Table 739. FRSR field descriptions

Field	Description
R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs. For proper operation, ensure that R_FSTART is set to 48h or greater.

48.4.21 Receive Descriptor Ring Start Register (ERDSR)

ERDSR points to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions.

This register is undefined at reset and must be initialized prior to operation.

Offset: 180h 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	R_DES_START																														0	0
W																																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 722. Receive Descriptor Ring Start Register (ERDSR)

Table 740. ERDSR field descriptions

Field	Description
R_DES_START	Pointer to start of receive buffer descriptor queue.

48.4.22 Transmit Buffer Descriptor Ring Start Register (ETSDR)

ETSDR provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions.

This register is undefined at reset and must be initialized prior to operation.

Offset: 184h 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	X_DES_START																														0	0
W																																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 723. Transmit Buffer Descriptor Ring Start Register (ETDSR)

Table 741. ETDSR field descriptions

Field	Description
X_DES_START	Pointer to start of transmit buffer descriptor queue.

48.4.23 Receive Buffer Size Register (EMRBR)

The EMRBR dictates the maximum size of all receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, EMRBR must be set to RCR[MAX_FL] or larger. To properly align the buffer, EMRBR must be evenly divisible by 16. To ensure this, bits 3–0 are forced low.

To minimize bus utilization (descriptor fetches), it is recommended that EMRBR be greater than or equal to 256 bytes.

The EMRBR is undefined at reset and must be initialized by the user.

Offset: 188h 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	R_BUF_SIZE							0	0	0	0
W																																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 724. Receive Buffer Size Register (EMRBR)

Table 742. EMRBR field descriptions

Field	Description
R_BUF_SIZE	Maximum size of receive buffer size in bytes. To minimize bus utilization (descriptor fetches), set this field to 256 bytes (10h) or larger. 10h 256 + 15 bytes (minimum size recommended) 11h 272 + 15 bytes ... 7Fh 2032 + 15 bytes. The FEC writes up to 2047 bytes in the receive buffer. If data larger than 2047 is received, the FEC truncates it and shows 7FFh in the receive descriptor

48.4.24 Count of frames not counted correctly (RMON_T_DROP)

Offset: 200h Access: Read-only

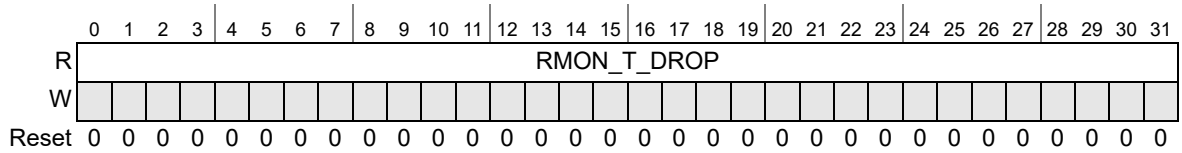


Figure 725. Count of frames not counted correctly (RMON_T_DROP)

Table 743. RMON_T_DROP field descriptions

Field	Description
RMON_T_DROP	Count of frames not counted correctly

48.4.25 RMON Tx packet count (RMON_T_PACKETS)

Offset: 204h Access: Read-only

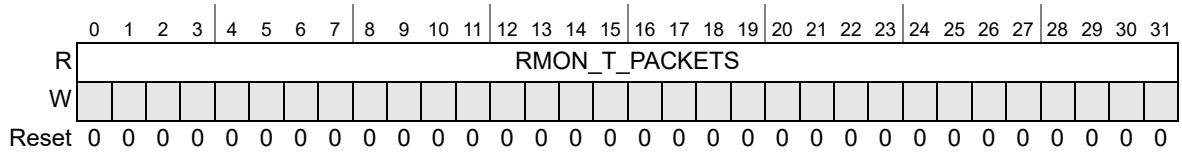


Figure 726. RMON Tx packet count (RMON_T_PACKETS)

Table 744. RMON_T_PACKETS field descriptions

Field	Description
RMON_T_PACKETS	RMON Tx packet count

48.4.26 RMON Tx broadcast packets (RMON_T_BC_PKT)

Offset: 208h Access: Read-only

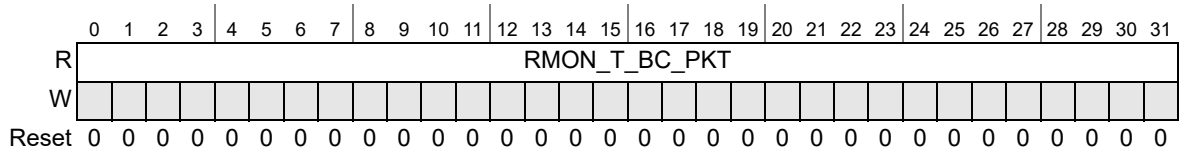


Figure 727. RMON Tx broadcast packets (RMON_T_BC_PKT)

Table 745. RMON_T_PACKETS field descriptions

Field	Description
RMON_T_BC_PKT	RMON Tx broadcast packets

48.4.27 RMON Tx multicast packets (RMON_T_MC_PKT)

Offset: 20Ch Access: Read-only

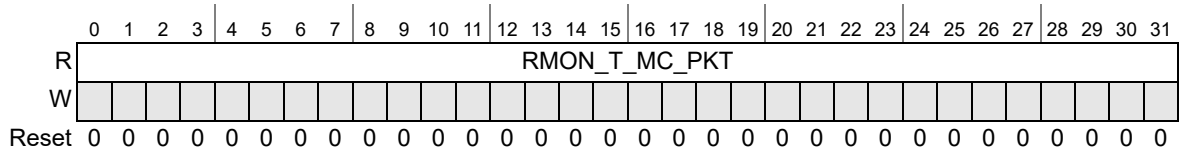


Figure 728. RMON Tx multicast packets (RMON_T_MC_PKT)

Table 746. RMON_T_MC_PKT field descriptions

Field	Description
RMON_T_MC_PKT	RMON Tx multicast packets

48.4.28 RMON Tx packets with CRC/align error (RMON_T_CRC_ALIGN)

Offset: 210h Access: Read-only

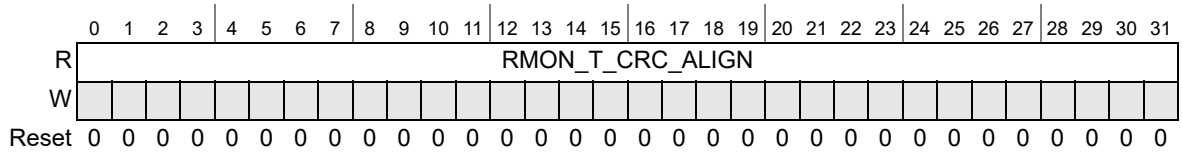


Figure 729. RMON Tx packets with CRC/align error (RMON_T_CRC_ALIGN)

Table 747. RMON_T_CRC_ALIGN field descriptions

Field	Description
RMON_T_CRC_ALIGN	RMON Tx packets with CRC/align error

48.4.29 RMON Tx packets < 64 bytes, good CRC (RMON_T_UNDERSIZE)

Offset: 214h Access: Read-only

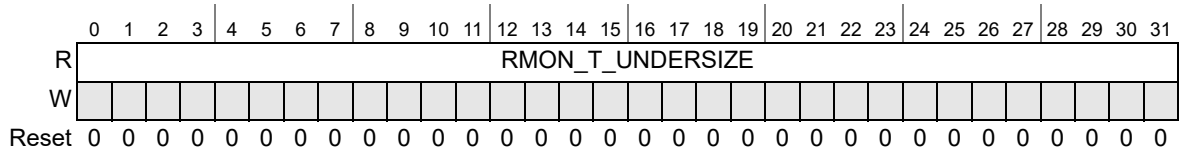


Figure 730. RMON Tx packets < 64 bytes, good CRC (RMON_T_UNDERSIZE)

Table 748. RMON_T_CRC_ALIGN field descriptions

Field	Description
RMON_T_UNDERSIZE	RMON Tx packets < 64 bytes, good CRC

48.4.30 RMON Tx packets > MAX_FL bytes, good CRC (RMON_T_OVERSIZE)

Offset: 218h Access: Read-only

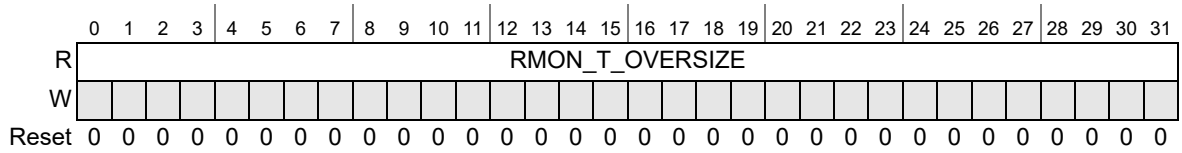


Figure 731. RMON Tx packets > MAX_FL bytes, good CRC (RMON_T_OVERSIZE)

Table 749. RMON_T_OVERSIZE field descriptions

Field	Description
RMON_T_OVERSIZE	RMON Tx packets > MAX_FL bytes, good CRC

48.4.31 RMON Tx packets < 64 bytes, bad CRC (RMON_T_FRAG)

Offset: 21Ch Access: Read-only

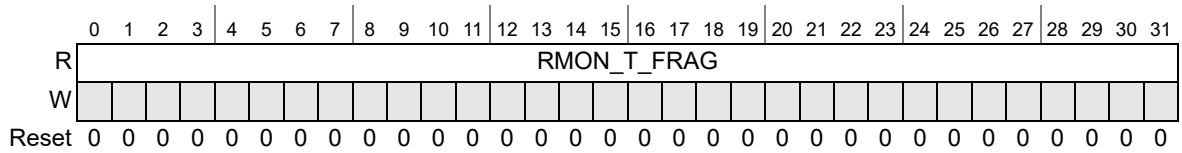


Figure 732. RMON Tx packets < 64 bytes, bad CRC (RMON_T_FRAG)

Table 750. RMON_T_FRAG field descriptions

Field	Description
RMON_T_FRAG	RMON Tx packets < 64 bytes, bad CRC

48.4.32 RMON Tx packets > MAX_FL bytes, bad CRC (RMON_T_JAB)

Offset: 220h Access: Read-only

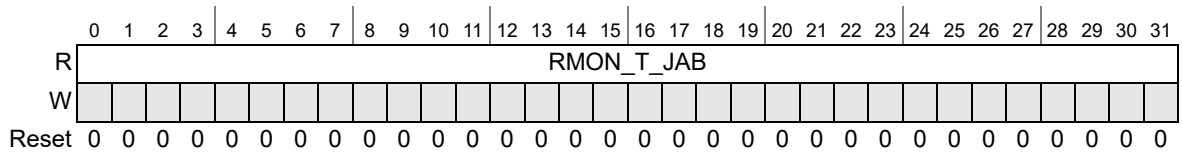


Figure 733. RMON Tx packets > MAX_FL bytes, bad CRC (RMON_T_JAB)

Table 751. RMON_T_JAB field descriptions

Field	Description
RMON_T_JAB	RMON Tx packets > MAX_FL bytes, bad CRC

48.4.33 RMON Tx collision count (RMON_T_COL)

Offset: 224h Access: Read-only

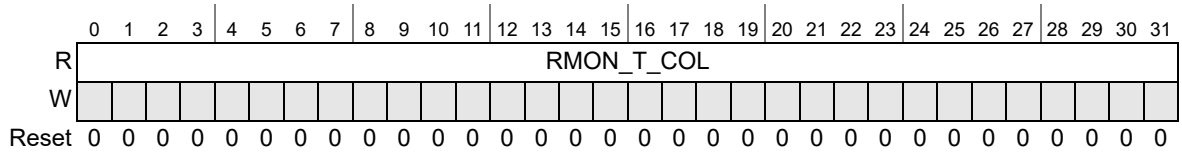


Figure 734. RMON Tx collision count (RMON_T_COL)

Table 752. RMON_T_COL field descriptions

Field	Description
RMON_T_COL	RMON Tx collision count

48.4.34 RMON Tx 64 byte packets (RMON_T_P64)

Offset: 228h Access: Read-only

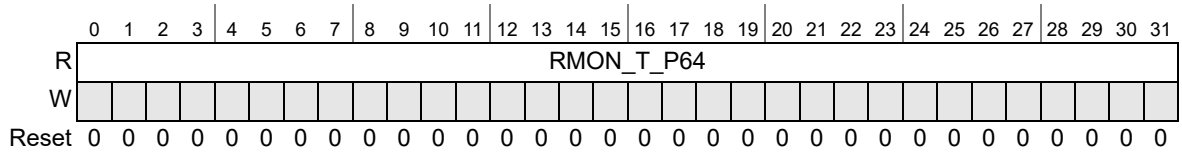


Figure 735. RMON Tx 64 byte packets (RMON_T_P64)

Table 753. RMON_T_P64 field descriptions

Field	Description
RMON_T_P64	RMON Tx 64 byte packets

48.4.35 RMON Tx 65 to 127 byte packets (RMON_T_P65TO127)

Offset: 22Ch Access: Read-only

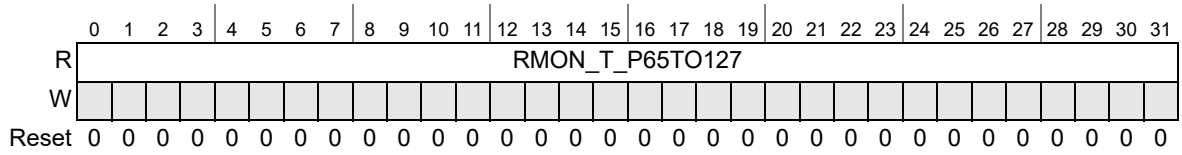


Figure 736. RMON Tx 65 to 127 byte packets (RMON_T_P65TO127)

Table 754. RMON_T_P65TO127 field descriptions

Field	Description
RMON_T_P65TO127	RMON Tx 65 to 127 byte packets

48.4.36 RMON Tx 128 to 255 byte packets (RMON_T_P128TO255)

Offset: 230h Access: Read-only

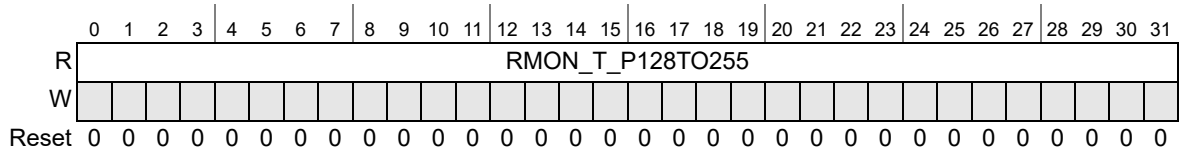


Figure 737. RMON Tx 128 to 255 byte packets (RMON_T_P128TO255)

Table 755. RMON_T_P128TO255 field descriptions

Field	Description
RMON_T_P128TO255	RMON Tx 128 to 255 byte packets

48.4.37 RMON Tx 256 to 511 byte packets (RMON_T_P256TO511)

Offset: 234h Access: Read-only

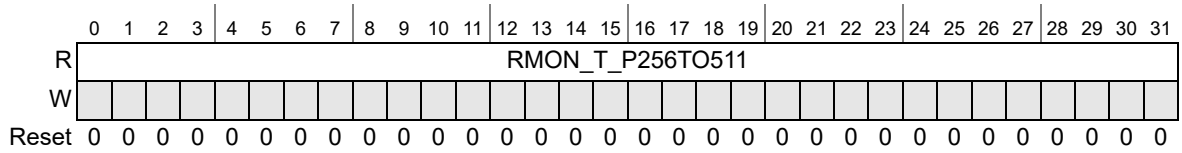


Figure 738. RMON Tx 256 to 511 byte packets (RMON_T_P256TO511)

Table 756. RMON_T_P256TO511 field descriptions

Field	Description
RMON_T_P256TO511	RMON Tx 256 to 511 byte packets

48.4.38 RMON Tx 512 to 1023 byte packets (RMON_T_P512TO1023)

Offset: 238h Access: Read-only

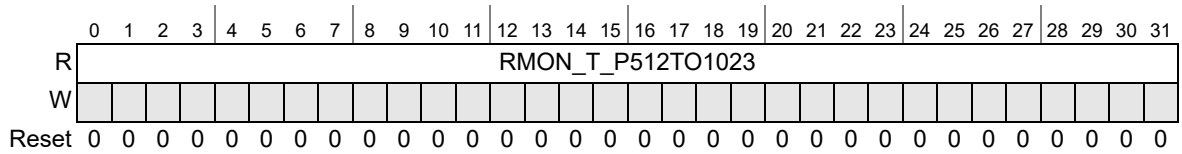


Figure 739. RMON Tx 512 to 1023 byte packets (RMON_T_P512TO1023)

Table 757. RMON_T_P512TO1023 field descriptions

Field	Description
RMON_T_P512TO1023	RMON Tx 512 to 1023 byte packets

48.4.39 RMON Tx 1024 to 2047 byte packets (RMON_T_P1024TO2047)

Offset: 23Ch 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	RMON_T_P1024TO2047																															
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 740. RMON Tx 1024 to 2047 byte packets (RMON_T_P1024TO2047)

Table 758. RMON_T_P1024TO2047 field descriptions

Field	Description
RMON_T_P1024TO2047	RMON Tx 1024 to 2047 byte packets

48.4.40 RMON Tx packets with > 2048 bytes (RMON_T_P_GTE2048)

Offset: 240h 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	RMON_T_P_GTE2048																															
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 741. RMON Tx packets with > 2048 bytes (RMON_T_P_GTE2048)

Table 759. RMON_T_P_GTE2048 field descriptions

Field	Description
RMON_T_P_GTE2048	RMON Tx packets with > 2048 bytes

48.4.41 RMON Tx Octets (RMON_T_OCTETS)

Offset: 244h 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	RMON_T_OCTETS																															
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 742. RMON Tx Octets (RMON_T_OCTETS)

Table 760. RMON_T_OCTETS field descriptions

Field	Description
RMON_T_OCTETS	RMON Tx Octets

48.4.42 Count of transmitted frames not counted correctly (IEEE_T_DROP)

Offset: 248h Access: Read-only

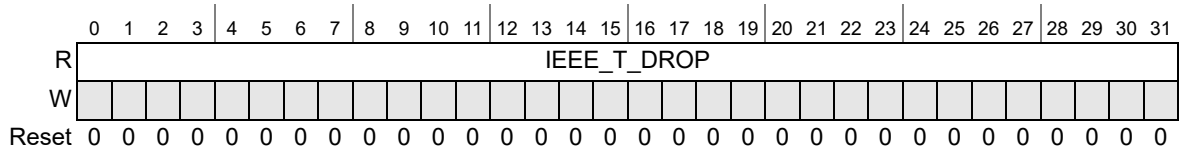


Figure 743. Count of transmitted frames not counted correctly (IEEE_T_DROP)

Table 761. IEEE_T_DROP field descriptions

Field	Description
IEEE_T_DROP	Count of transmitted frames not counted correctly

48.4.43 Frames transmitted OK (IEEE_T_FRAME_OK)

Offset: 24Ch Access: Read-only

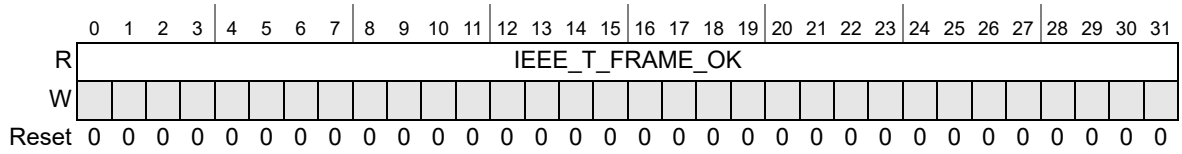


Figure 744. Frames transmitted OK (IEEE_T_FRAME_OK)

Table 762. IEEE_T_FRAME_OK field descriptions

Field	Description
IEEE_T_FRAME_OK	Frames transmitted OK

48.4.44 Frames transmitted with single collision (IEEE_T_1COL)

Offset: 250h Access: Read-only

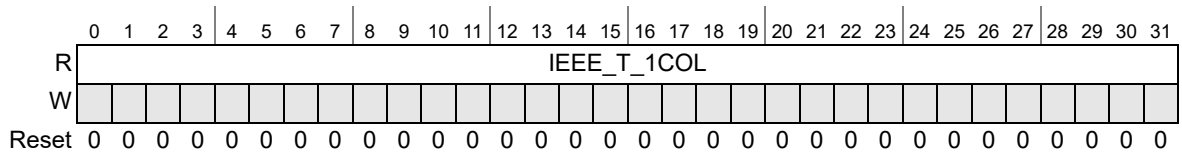


Figure 745. Frames transmitted with single collision (IEEE_T_1COL)

Table 763. IEEE_T_1COL field descriptions

Field	Description
IEEE_T_1COL	Frames transmitted with single collision

48.4.45 Frames transmitted with multiple collisions (IEEE_T_MCOL)

Offset: 254h Access: Read-only

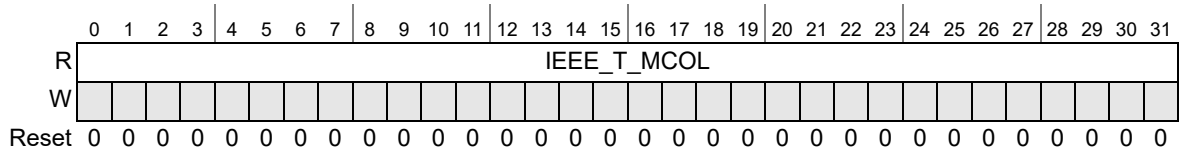


Figure 746. Frames transmitted with multiple collisions (IEEE_T_MCOL)

Table 764. IEEE_T_MCOL field descriptions

Field	Description
IEEE_T_MCOL	Frames transmitted with multiple collisions

48.4.46 Frames transmitted after deferral delay (IEEE_T_DEF)

Offset: 258h Access: Read-only

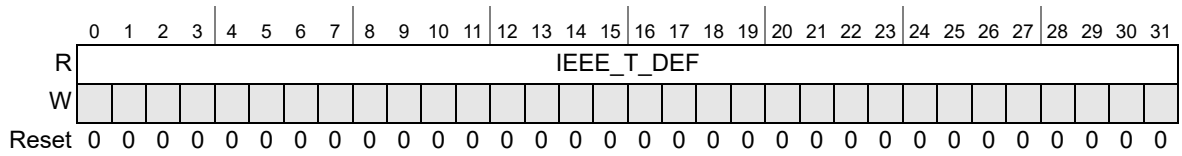


Figure 747. Frames transmitted after deferral delay (IEEE_T_DEF)

Table 765. IEEE_T_DEF field descriptions

Field	Description
IEEE_T_DEF	Frames transmitted after deferral delay

48.4.47 Frames transmitted with late collision (IEEE_T_LCOL)

Offset: 25Ch Access: Read-only

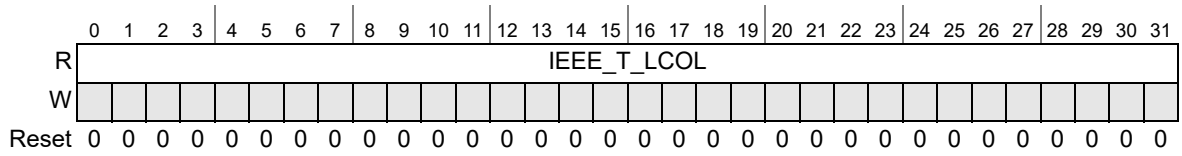


Figure 748. Frames transmitted with late collision (IEEE_T_LCOL)

Table 766. IEEE_T_LCOL field descriptions

Field	Description
IEEE_T_LCOL	Frames transmitted with late collision

48.4.48 Frames transmitted with excessive collisions (IEEE_T_EXCOL)

Offset: 260h 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	IEEE_T_EXCOL																															
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 749. Frames transmitted with excessive collisions (IEEE_T_EXCOL)

Table 767. IEEE_T_EXCOL field descriptions

Field	Description
IEEE_T_EXCOL	Frames transmitted with excessive collisions

48.4.49 Frames transmitted with Tx FIFO underrun (IEEE_T_MACERR)

Offset: 264h 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	IEEE_T_MACERR																															
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 750. Frames transmitted with Tx FIFO underrun (IEEE_T_MACERR)

Table 768. IEEE_T_MACERR field descriptions

Field	Description
IEEE_T_MACERR	Frames transmitted with Tx FIFO underrun

48.4.50 Frames transmitted with carrier sense error (IEEE_T_CSERR)

Offset: 268h 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	IEEE_T_CSERR																															
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 751. Frames transmitted with carrier sense error (IEEE_T_CSERR)

Table 769. IEEE_T_CSERR field descriptions

Field	Description
IEEE_T_CSERR	Frames transmitted with carrier sense error

48.4.51 Frames transmitted with SQE error (IEEE_T_SQE)

Offset: 26Ch 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	IEEE_T_SQE																															
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 752. Frames transmitted with SQE error (IEEE_T_SQE)

Table 770. IEEE_T_SQE field descriptions

Field	Description
IEEE_T_SQE	Frames transmitted with SQE error

48.4.52 Flow control pause frames transmitted (IEEE_T_FDXFC)

Offset: 270h 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	IEEE_T_FDXFC																															
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 753. Flow control pause frames transmitted (IEEE_T_FDXFC)

Table 771. IEEE_T_FDXFC field descriptions

Field	Description
IEEE_T_FDXFC	Flow control pause frames transmitted

48.4.53 Octet count for frames transmitted without error (IEEE_T_OCTETS_OK)

Offset: 274h 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	IEEE_T_OCTETS_OK																															
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 754. Octet count for frames transmitted without error (IEEE_T_OCTETS_OK)

Table 772. IEEE_T_OCTETS_OK field descriptions

Field	Description
IEEE_T_OCTETS_OK	Octet count for frames transmitted without error

48.4.54 Count of received frames not counted correctly (RMON_R_DROP)

Offset: 280h Access: Read-only

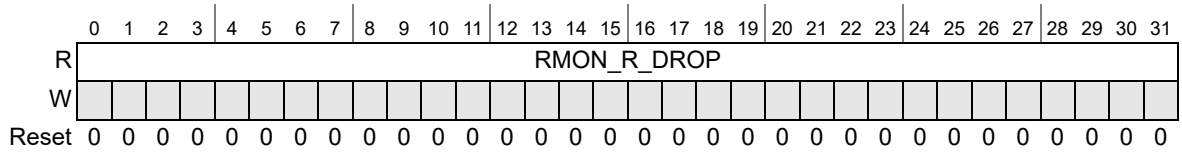


Figure 755. Count of received frames not counted correctly (RMON_R_DROP)

Table 773. RMON_R_DROP field descriptions

Field	Description
RMON_R_DROP	Count of received frames not counted correctly

48.4.55 RMON Rx packet count (RMON_R_PACKETS)

Offset: 284h Access: Read-only

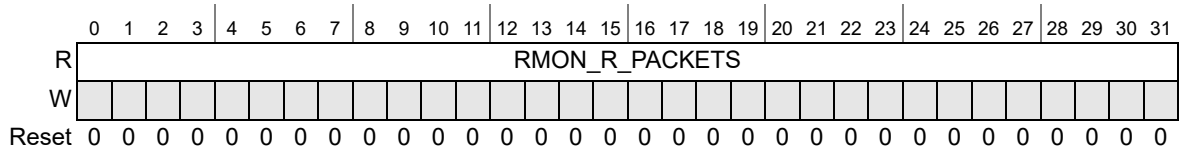


Figure 756. RMON Rx packet count (RMON_R_PACKETS)

Table 774. RMON_R_PACKETS field descriptions

Field	Description
RMON_R_PACKETS	RMON Rx packet count

48.4.56 RMON Rx broadcast packets (RMON_R_BC_PKT)

Offset: 288h Access: Read-only

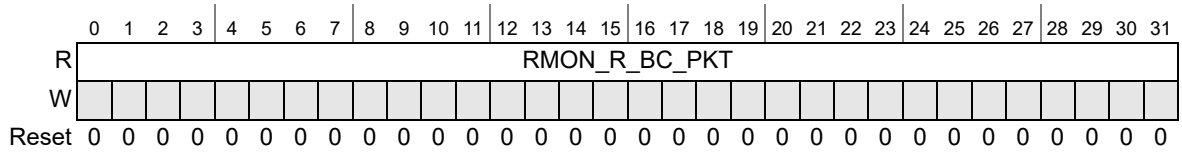


Figure 757. RMON Rx broadcast packets (RMON_R_BC_PKT)

Table 775. RMON_R_BC_PKT field descriptions

Field	Description
RMON_R_BC_PKT	RMON Rx broadcast packets

48.4.57 RMON Rx multicast packets (RMON_R_MC_PKT)

Offset: 28Ch Access: Read-only

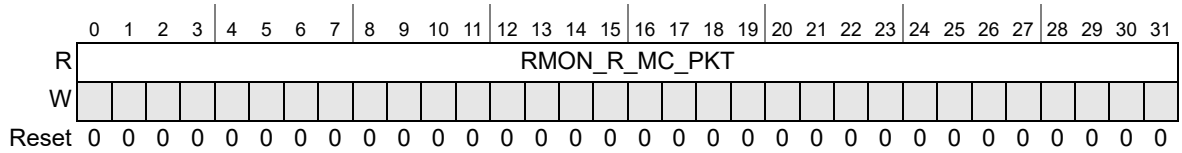


Figure 758. RMON Rx multicast packets (RMON_R_MC_PKT)

Table 776. RMON_R_MC_PKT field descriptions

Field	Description
RMON_R_MC_PKT	RMON Rx multicast packets

48.4.58 RMON Rx packets with CRC/Align error (RMON_R_CRC_ALIGN)

Offset: 290h Access: Read-only

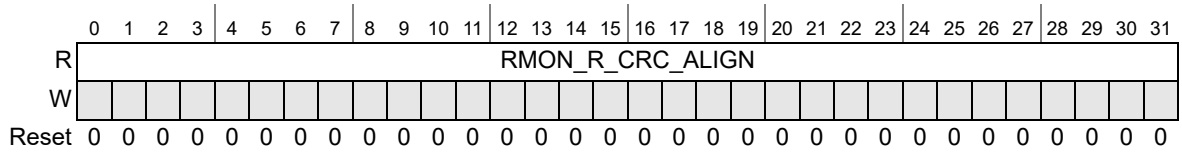


Figure 759. RMON Rx packets with CRC/Align error (RMON_R_CRC_ALIGN)

Table 777. RMON_R_CRC_ALIGN field descriptions

Field	Description
RMON_R_CRC_ALIGN	RMON Rx packets with CRC/Align error

48.4.59 RMON Rx packets < 64 bytes, good CRC (RMON_R_UNDERSIZE)

Offset: 294h Access: Read-only

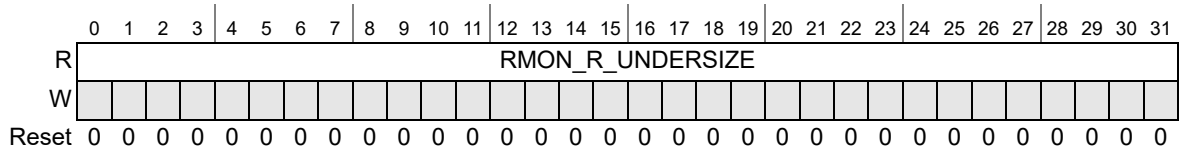


Figure 760. RMON Rx packets < 64 bytes, good CRC (RMON_R_UNDERSIZE)

Table 778. RMON_R_UNDERSIZE field descriptions

Field	Description
RMON_R_UNDERSIZE	RMON Rx packets < 64 bytes, good CRC

48.4.60 RMON Rx packets > MAX_FL bytes, good CRC (RMON_R_OVERSIZE)

Offset: 298h Access: Read-only

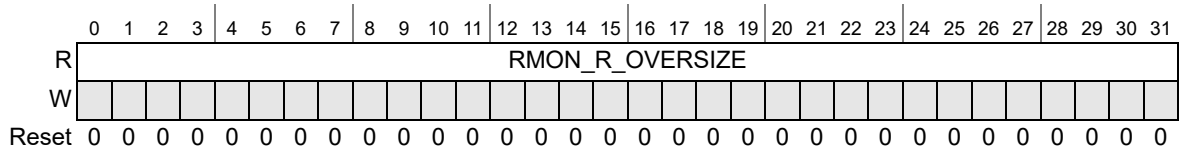


Figure 761. RMON Rx packets > MAX_FL bytes, good CRC (RMON_R_OVERSIZE)

Table 779. RMON_R_OVERSIZE field descriptions

Field	Description
RMON_R_OVERSIZE	RMON Rx packets > MAX_FL bytes, good CRC

48.4.61 RMON Rx packets < 64 bytes, bad CRC (RMON_R_FRAG)

Offset: 29Ch Access: Read-only

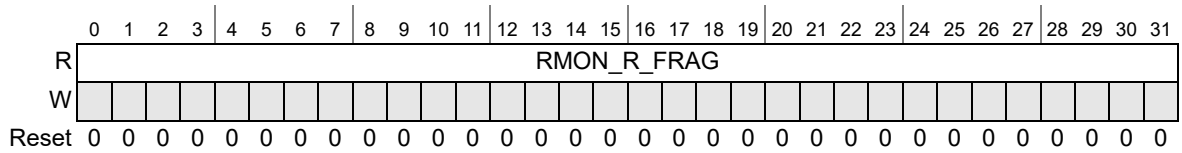


Figure 762. RMON Rx packets < 64 bytes, bad CRC (RMON_R_FRAG)

Table 780. RMON_R_FRAG field descriptions

Field	Description
RMON_R_FRAG	RMON Rx packets < 64 bytes, bad CRC

48.4.62 RMON Rx packets > MAX_FL bytes, bad CRC (RMON_R_JAB)

Offset: 2A0h Access: Read-only

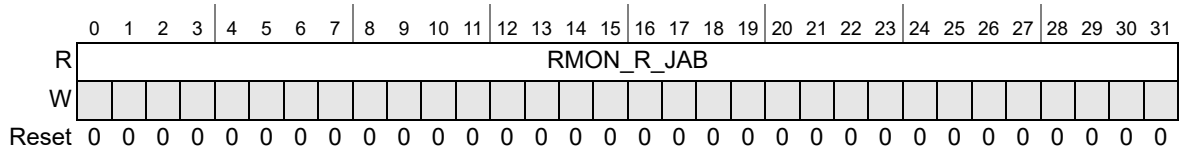


Figure 763. RMON Rx packets > MAX_FL bytes, bad CRC (RMON_R_JAB)

Table 781. RMON_R_JAB field descriptions

Field	Description
RMON_R_JAB	RMON Rx packets > MAX_FL bytes, bad CRC

48.4.63 Reserved (RMON_R_RESVD_0)

Offset: 2A4h Access: Read-only

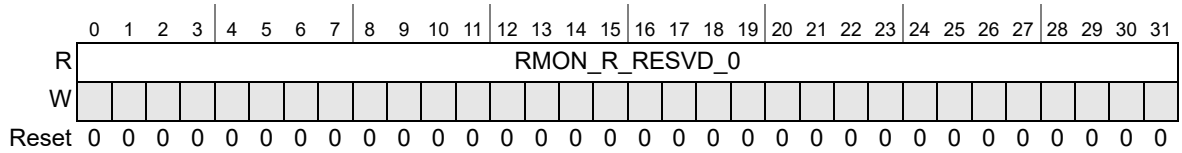


Figure 764. Reserved (RMON_R_RESVD_0)

Table 782. RMON_R_RESVD_0 field descriptions

Field	Description
RMON_R_RESVD_0	Reserved

48.4.64 RMON Rx 64 byte packets (RMON_R_P64)

Offset: 2A8h Access: Read-only

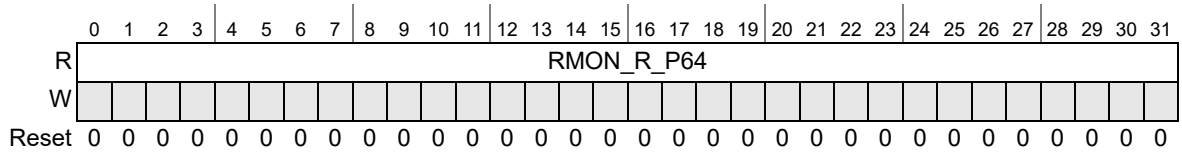


Figure 765. RMON Rx 64 byte packets (RMON_R_P64)

Table 783. RMON_R_P64 field descriptions

Field	Description
RMON_R_P64	RMON Rx 64 byte packets

48.4.65 RMON Rx 65 to 127 byte packets (RMON_R_P65TO127)

Offset: 2ACh Access: Read-only

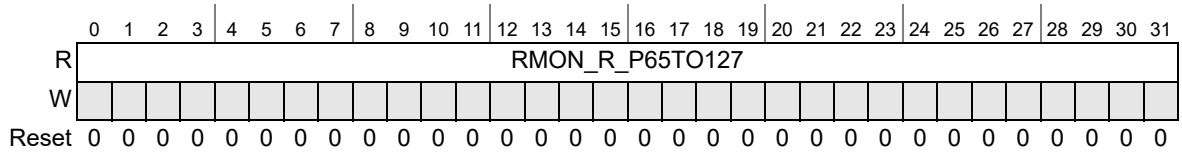


Figure 766. RMON Rx 65 to 127 byte packets (RMON_R_P65TO127)

Table 784. RMON_R_P65TO127 field descriptions

Field	Description
RMON_R_P65TO127	RMON Rx 65 to 127 byte packets

48.4.66 RMON Rx 128 to 255 byte packets (RMON_R_P128TO255)

Offset: 2B0h 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	RMON_R_P128TO255																															
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 767. RMON Rx 128 to 255 byte packets (RMON_R_P128TO255)

Table 785. RMON_R_P128TO255 field descriptions

Field	Description
RMON_R_P128TO255	RMON Rx 128 to 255 byte packets

48.4.67 RMON Rx 256 to 511 byte packets (RMON_R_P256TO511)

Offset: 2B4h 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	RMON_R_P256TO511																															
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 768. RMON Rx 256 to 511 byte packets (RMON_R_P256TO511)

Table 786. RMON_R_P256TO511 field descriptions

Field	Description
RMON_R_P256TO511	RMON Rx 256 to 511 byte packets

48.4.68 RMON Rx 512 to 1023 byte packets (RMON_R_P512TO1023)

Offset: 2B8h 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	RMON_R_P512TO1023																															
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 769. RMON Rx 512 to 1023 byte packets (RMON_R_P512TO1023)

Table 787. RMON_R_P512TO1023 field descriptions

Field	Description
RMON_R_P512TO 1023	RMON Rx 512 to 1023 byte packets

48.4.69 RMON Rx 1024 to 2047 byte packets (RMON_R_P1024TO2047)

Offset: 2BCh 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	RMON_R_P1024TO2047																															
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 770. RMON Rx 1024 to 2047 byte packets (RMON_R_P1024TO2047)

Table 788. RMON_R_P1024TO2047 field descriptions

Field	Description
RMON_R_P1024TO2047	RMON Rx 1024 to 2047 byte packets

48.4.70 RMON Rx packets with > 2048 bytes (RMON_R_P_GTE2048)

Offset: 2C0h 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	RMON_R_P_GTE2048																															
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 771. RMON Rx packets with > 2048 bytes (RMON_R_P_GTE2048)

Table 789. RMON_R_P_GTE2048 field descriptions

Field	Description
RMON_R_P_GTE2048	RMON Rx packets with > 2048 bytes

48.4.71 RMON Rx octets (RMON_R_OCTETS)

Offset: 2C4h 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	RMON_R_OCTETS																															
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 772. RMON Rx octets (RMON_R_OCTETS)

Table 790. RMON_R_OCTETS field descriptions

Field	Description
RMON_R_OCTETS	RMON Rx octets

48.4.72 Count of received frames not counted correctly (IEEE_R_DROP)

Offset: 2C8h Access: Read-only

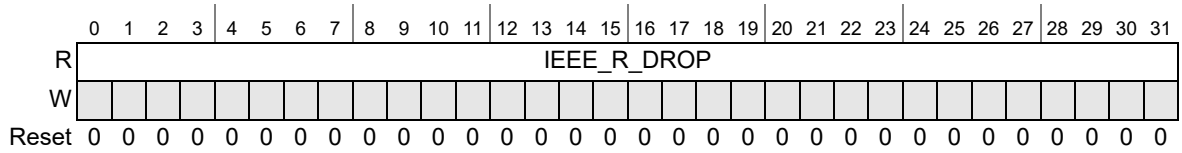


Figure 773. Count of received frames not counted correctly (IEEE_R_DROP)

Table 791. IEEE_R_DROP field descriptions

Field	Description
IEEE_R_DROP	Count of received frames not counted correctly

48.4.73 Frames received OK (IEEE_R_FRAME_OK)

Offset: 2CCh Access: Read-only

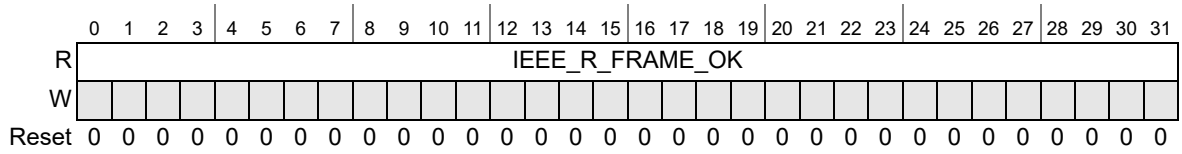


Figure 774. Frames received OK (IEEE_R_FRAME_OK)

Table 792. IEEE_R_FRAME_OK field descriptions

Field	Description
IEEE_R_FRAME_OK	Frames received OK

48.4.74 Frames received with CRC error (IEEE_R_CRC)

Offset: 2D0h Access: Read-only

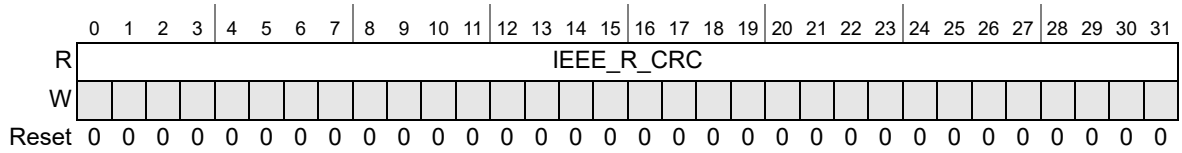


Figure 775. Frames received with CRC error (IEEE_R_CRC)

Table 793. IEEE_R_CRC field descriptions

Field	Description
IEEE_R_CRC	Frames received with CRC error

48.4.75 Frames received with alignment error (IEEE_R_ALIGN)

Offset: 2D4h Access: Read-only

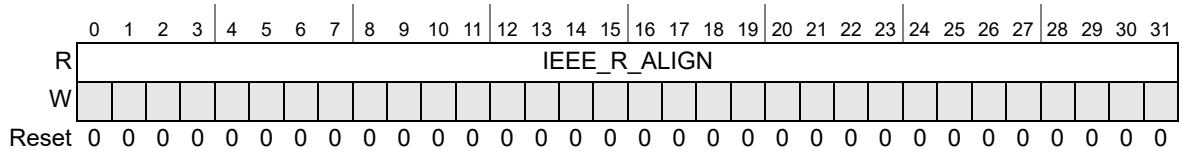


Figure 776. Frames received with alignment error (IEEE_R_ALIGN)

Table 794. IEEE_R_ALIGN field descriptions

Field	Description
IEEE_R_ALIGN	Frames received with alignment error

48.4.76 Receive FIFO overflow count (IEEE_R_MACERR)

Offset: 2D8h Access: Read-only

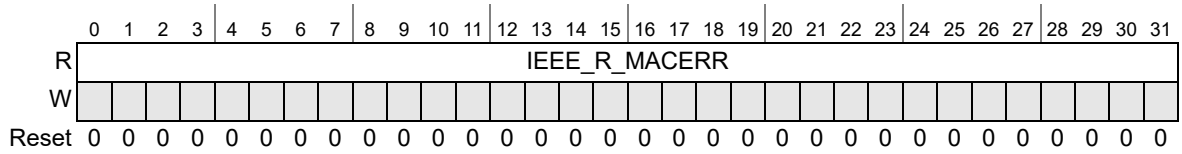


Figure 777. Receive FIFO overflow count (IEEE_R_MACERR)

Table 795. IEEE_R_MACERR field descriptions

Field	Description
IEEE_R_MACERR	Receive FIFO overflow count

48.4.77 Flow control pause frames received (IEEE_R_FDXFC)

Offset: 2DCh Access: Read-only

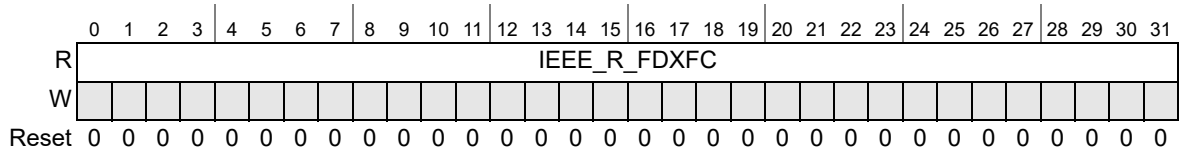


Figure 778. Flow control pause frames received (IEEE_R_FDXFC)

Table 796. IEEE_R_FDXFC field descriptions

Field	Description
IEEE_R_FDXFC	Flow control pause frames received

48.4.78 Octet count for frames received without error (IEEE_R_OCTETS_OK)

Offset: 2E0h 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	IEEE_R_OCTETS_OK																															
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 779. Octet count for frames received without error (IEEE_R_OCTETS_OK)

Table 797. IEEE_R_OCTETS_OK field descriptions

Field	Description
IEEE_R_OCTETS_OK	Octet count for frames received without error

48.5 Functional description

This section describes the operation of the FEC, beginning with the description of two different types of MII frames, the buffer descriptors, the hardware and software initialization sequence, then the software (Ethernet driver) interface for transmitting and receiving frames.

Following the software initialization and operation sections are sections providing a detailed description of the functions of the FEC.

48.5.1 MII data frame

Ethernet/802.3 data frames transmitted across the MII have the format specified in *Figure 780* and *Table 798*.

<inter-frame> <preamble> <sfd> <data> <efd>

Figure 780. MII data frame format

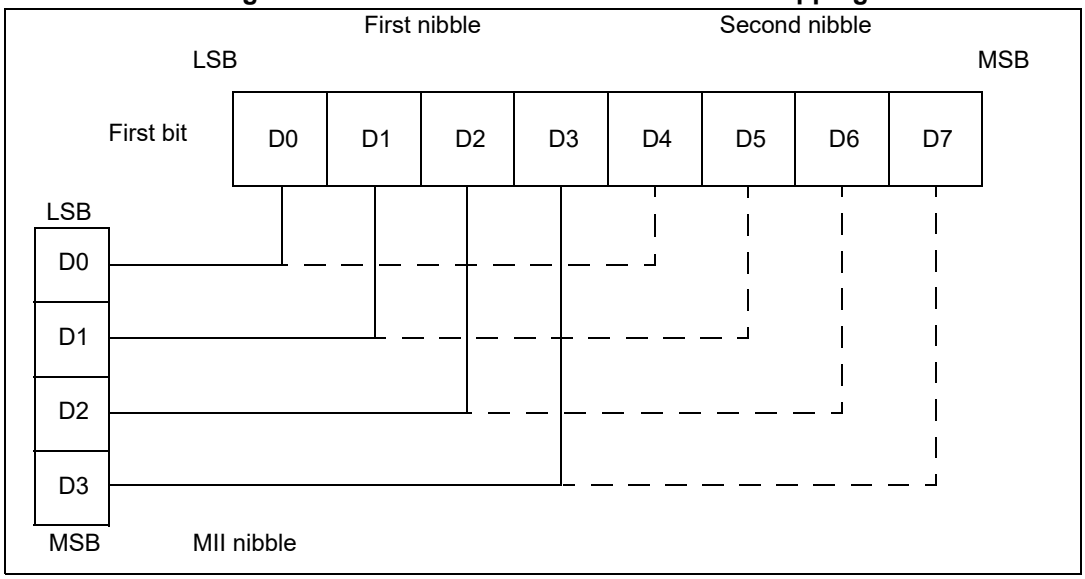
Table 798. MII data frame components

Component	Description
<inter-frame>	Inter-frame period. This is an unspecified amount of time during which no data activity occurs on the MII. The de-assertion of RX_DV and TX_EN indicate the absence of data activity.
<preamble>	Preamble that begins a frame. It has the following value: 10101010 10101010 10101010 10101010 10101010 10101010 10101010 The leftmost 1 represents the LSB of the byte.
<sfd>	Start of a frame. It has the value 10101011.

Table 798. MII data frame components(Continued)

Component	Description
<data>	Data portion of the frame. It consists of N octets which corresponds to 2N nibbles being transmitted. The order of each nibble is defined in *Figure 781*.
<efd>	End-of-frame delimiter. For TXD data, it is indicated by the de-assertion of the TX_EN signal. For RXD data, it is indicated by the de-assertion of the RX_DV signal.

Figure 781. MII nibble/octet to octet/nibble mapping



48.5.2 MII management frame structure

A transceiver management frame being transmitted on the MII management interface uses the MDIO and MDC pins. A transaction or frame on this serial interface has the format specified in *Figure 782* and *Table 799*.

<preamble> <st> <op> <phyad> <regad> <ta> <data> <idle>

Figure 782. MII management frame format

Table 799. MII management frame components

Component	Description
<preamble>	Optional preamble, consisting of a sequence of 32 continuous logic 1's.
<st>	Start of frame, consisting of a 01 pattern.
<op>	Operation code. 01 Write operation 10 Read instruction
<phyad>	Five-bit field that allows up to 32 PHYs to be addressed. The first address bit transmitted is the MSB of the address.

Table 799. MII management frame components(Continued)

Component	Description
<regad>	Five-bit field that allows for 32 registers to be addressed within each PHY. The first register bit transmitted is the MSB of the address.
<ta>	Two-bit field that provides spacing between the register address field and the data field to avoid contention on the MDIO signal during a read operation.
<data>	16-bit data field. Bit 15 is the first data bit transmitted and received.
<idle>	Idle condition (MDIO is in the high-impedance state).

The MII management register set located in the PHY may consist of a basic register set and an extended register set as defined in *Table 800*.

Table 800. MII management register set

Register address	Register name	Register set
0	Control	Basic
1	Status	
2–3	PHY identifier	Extended
4	Auto-negotiation advertisement	
5	AN link partner ability	
6	AN expansion	
7	AN next page transmit	
8–15	Reserved	
16–31	Vendor-specific	

48.5.3 Buffer descriptors

This section provides a description of the operation of the driver/DMA via the buffer descriptors. It is followed by a detailed description of the receive and transmit descriptor fields.

48.5.3.1 Driver/DMA operation with buffer descriptors

The data for the FEC frames resides in one or more memory buffers external to the FEC. Associated with each buffer is a buffer descriptor (BD), which contains a starting address (32-bit aligned pointer), data length, and status/control information (which contains the current state for the buffer). To permit maximum user flexibility, the BDs are also located in external memory and are read by the FEC DMA engine.

Software produces buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] bit produces the buffer. Software writing to TDAR or RDAR tells the FEC that a buffer is placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and consumes the buffers after they have been produced. After the data DMA is complete and the DMA engine writes the buffer descriptor status bits, hardware clears RxBD[E] or TxBD[R] to signal the buffer has been consumed. Software may poll the BDs to detect when the buffers are consumed

or may rely on the buffer/frame interrupts. The driver may process these buffers, and they can return to the free list.

The ECR[ETHER_EN] bit operates as a reset to the BD/DMA logic. When ECR[ETHER_EN] is cleared, the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software before ECR[ETHER_EN] is set.

The buffer descriptors operate as two separate rings. ERDSR defines the starting address for receive BDs and ETDSR defines the starting address for transmit BDs. The wrap (W) bit defines the last buffer descriptor in each ring. When W is set, the next descriptor in the ring is at the location pointed to by ERDSR and ETDSR for the receive and transmit rings, respectively. Buffer descriptor rings must start on a 32-bit boundary; however, it is recommended they be 128-bit aligned.

48.5.3.1.1 Driver/DMA operation with transmit BDs

Typically, a transmit frame is divided between multiple buffers. An example is to have an application payload in one buffer, TCP header in a second buffer, IP header in a third buffer, and Ethernet/IEEE 802.3 header in a fourth buffer. The Ethernet MAC does not prepend the Ethernet header (destination address, source address, length/type field(s)), so the driver must provide this in one of the transmit buffers. The Ethernet MAC can append the Ethernet CRC to the frame. TxBD[TC], which must be set by the driver, determines whether the MAC or driver appends the CRC.

The driver (TxBD software producer) should set up Tx BDs so a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, the BDs should be initialized with pointer, length, and control (W, L, TC, ABC) and then the TxBD[R] bit should be set in reverse order (third, second, then first BD) to ensure that the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA controller could DMA the first BD before the second was made available, potentially causing a transmit FIFO underrun.

In the FEC, the driver notifies the DMA that new transmit frame(s) are available by writing to TDAR. When this register is written to (data value is not significant), the FEC RISC tells the DMA to read the next transmit BD in the ring. After started, the RISC + DMA continues to read and interpret transmit BDs in order and DMA the associated buffers until a transmit BD is encountered with the R bit cleared. At this point, the FEC polls this BD one more time. If the R bit is cleared the second time, RISC stops the transmit descriptor read process until software sets up another transmit frame and writes to TDAR.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R bit, indicating that the hardware consumer is finished with the buffer.

48.5.3.1.2 Driver/DMA operation with receive BDs

Unlike transmit, the length of the receive frame is unknown by the driver ahead of time. Therefore, the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the EMRBR register.

The driver (RxBD software producer) should set up some number of empty buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) consumes these buffers by filling them with data as frames are received and clearing the E bit and writing to the L bit (1 indicates last buffer in frame), the frame status bits (if L is set), and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For non-last buffers, the length field in the receive BD is written by the DMA (at the same time the E bit is cleared) with the default receive buffer length value. For end-of-frame buffers, the receive BD is written with L set and information written to the status bits (M, BC, MC, LG, NO, CR, OV, TR). Some of the status bits are error indicators which, if set, indicate the receive frame should be discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end of frame buffer is written with the length of the entire frame, not only the length of the last buffer.

For simplicity, the driver may assign a large enough default receive buffer length to contain an entire frame, keeping in mind that a malfunction on the network or out-of-spec implementation could result in giant frames. Frames of 2K (2048) bytes or larger are truncated by the FEC at 2047 bytes so software never sees a receive frame larger than 2047 bytes.

Similar to transmit, the FEC polls the receive descriptor ring after the driver sets up receive BDs and writes to the RDAR register. As frames are received, the FEC fills receive buffers and updates the associated BDs, then reads the next BD in the receive descriptor ring. If the FEC reads a receive BD and finds the E bit cleared, it polls this BD once more. If RxBD[E] is clear a second time, FEC stops reading receive BDs until the driver writes to RDAR.

48.5.3.2 Ethernet Receive Buffer Descriptor (RxBD)

In the RxBD, the user initializes the E and W bits in the first longword and the pointer in the second longword. When the buffer has been DMA'd, the Ethernet controller modifies the E, L, M, BC, MC, LG, NO, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first longword. The M, BC, MC, LG, NO, CR, OV, and TR bits in the first longword of the buffer descriptor are only modified by the Ethernet controller when the L bit is set.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data Length															
Offset + 4	Rx Data Buffer Pointer - A[31:16]															
Offset + 6	Rx Data Buffer Pointer - A[15:0]															

Figure 783. Receive Buffer Descriptor (RxBD)

Table 801. Receive Buffer Descriptor field descriptions

Word	Field	Description
Offset + 0	E	Empty. Written by the FEC (=0) and user (=1). 0 The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	RO1	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ERDSR.

Table 801. Receive Buffer Descriptor field descriptions(Continued)

Word	Field	Description
Offset + 0	RO2	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	L	Last in frame. Written by the FEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 0	M	Miss. Written by the FEC. This bit is set by the FEC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use the M-bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L-bit is set and the PROM bit is set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
Offset + 0	BC	Set if the DA is broadcast (FFFF_FFFF_FFFF).
Offset + 0	MC	Set if the DA is multicast and not BC.
Offset + 0	LG	Rx frame length violation. Written by the FEC. A frame length greater than RCR[MAX_FL] was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.
Offset + 0	NO	Receive non-octet aligned frame. Written by the FEC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set, the CR bit is not set.
Offset + 0	CR	Receive CRC error. Written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.
Offset + 0	OV	Overrun. Written by the FEC. A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR, and CL lose their normal meaning and are zero. This bit is valid only if the L-bit is set.
Offset + 0	TR	Set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect.
Offset + 2	Data Length	Data length. Written by the FEC. Data length is the number of octets written by the FEC into this BD's data buffer if L equals 0 (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the FEC once as the BD is closed.
0ffset + 4	A[31:16]	RX data buffer pointer, bits [31:16](1)
Offset + 6	A[15:0]	RX data buffer pointer, bits [15:0]

1. The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 16. The buffer must reside in memory external to the FEC. The Ethernet controller never modifies this value.

Note: When the software driver sets an E bit in one or more receive descriptors, the driver should follow with a write to RDAR.

48.5.3.3 Ethernet Transmit Buffer Descriptor (TxBD)

Data is presented to the FEC for transmission by arranging it in buffers referenced by the channel's TxBDs. The Ethernet controller confirms transmission by clearing the ready bit (TxBD[R]) when DMA of the buffer is complete. In the TxBD, the user initializes the R, W, L, and TC bits and the length (in bytes) in the first longword and the buffer pointer in the second longword.

The FEC clears the R bit when the buffer is transferred. Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated via individual interrupt bits (error conditions) and in statistic counters in the MIB block. See the memory map for more details.

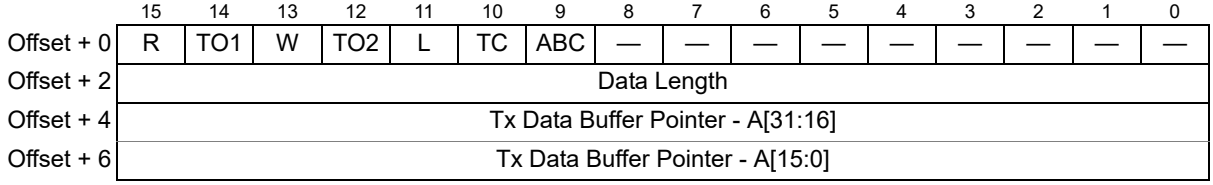


Figure 784. Transmit Buffer Descriptor (TxBD)

Table 802. Transmit Buffer Descriptor field descriptions

Word	Field	Description
Offset + 0	R	Ready. Written by the FEC and you. 0 The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this bit is set.
Offset + 0	TO1	Transmit software ownership. This field is reserved for software use. This read/write bit is not modified by hardware nor does its value affect hardware.
Offset + 0	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.
Offset + 0	TO2	Transmit software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware nor does its value affect hardware.
Offset + 0	L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame 1 The buffer is the last in the transmit frame
Offset + 0	TC	Transmit CRC. Written by user (only valid if L is set). 0 End transmission immediately after the last data byte 1 Transmit the CRC sequence after the last data byte
Offset + 0	ABC	Append bad CRC. Written by user (only valid if L is set). 0 No effect 1 Transmit the CRC sequence inverted after the last data byte (regardless of TC value)
Offset + 2	Data Length	Data length, written by user. Data length is the number of octets the FEC should transmit from this BD's data buffer. It is never modified by the FEC.
Offset + 4	A[31:16]	Tx data buffer pointer, bits [31:16](1)
Offset + 6	A[15:0]	Tx data buffer pointer, bits [15:0]

1. The transmit buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 4. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

Note: After the software driver has set up the buffers for a frame, it should set up the corresponding BDs. The last step in setting up the BDs for a transmit frame is setting the R

bit in the first BD for the frame. The driver must follow that with a write to TDAR that triggers the FEC to poll the next BD in the ring.

48.5.4 Initialization sequence

This section describes which registers are reset due to hardware reset, which registers are reset by the FEC RISC, and what locations you must initialize prior to enabling the FEC.

48.5.4.1 Hardware Controlled Initialization

In the FEC, hardware resets registers and control logic that generate interrupts. A hardware reset negates output signals and resets general configuration bits.

Other registers reset when the ECR[ETHER_EN] bit is cleared (which is accomplished by a hard reset or software to halt operation). By clearing ECR[ETHER_EN], configuration control registers such as the TCR and RCR are not reset, but the entire data path is reset.

Table 803. ECR[ETHER_EN] De-Assertion Effect on FEC

Register/Machine	Reset Value
XMIT block	Transmission is aborted (bad CRC appended)
RECV block	Receive activity is aborted
DMA block	All DMA activity is terminated
RDAR	Cleared
TDAR	Cleared
Descriptor Controller block	Halt operation

48.5.5 User initialization (Prior to Setting ECR[ETHER_EN])

You need to initialize portions the FEC prior to setting the ECR[ETHER_EN] bit. The exact values depend on the particular application. The sequence is not important.

The following registers require initialization:

- EIMR
- EIR (clear by writing FFFF_FFFFh)
- TFWR (optional)
- IALR / IAUR
- GAUR / GALR
- PALR / PAUR (only needed for full duplex flow control)
- OPD (only needed for full duplex flow control)
- RCR
- TCR
- MSCR (optional)
- MIB_RAM (clear)
- FRSR (optional)
- EMRBR
- ERDSR
- ETDSR
- Transmit Descriptor ring (empty)
- Receive Descriptor ring (empty)

48.5.6 Microcontroller initialization

In the FEC, the descriptor control RISC initializes some registers after ECR[ETHER_EN] is asserted. After the microcontroller initialization sequence is complete, hardware is ready for operation.

The sequence is as follows:

1. Initialize the backoff random number seed.
2. Activate receiver.
3. Activate transmitter.
4. Clear transmit FIFO.
5. Clear receive FIFO.
6. Initialize the transmit ring pointer.
7. Initialize the receive ring pointer.
8. Initialize FIFO count registers.

48.5.7 User initialization (after setting ECR[ETHER_EN])

After setting ECR[ETHER_EN], you can set up the buffer/frame descriptors and write to TDAR and RDAR. See *Section 48.5.3: Buffer descriptors*, for more details.

48.5.8 Network interface options

The FEC supports an MII and reduced MII interface for 10/100 Mbps Ethernet, as well as a 7-wire serial interface for 10 Mbps Ethernet. The RCR[MII_MODE] and RCR[RMII_MODE] bits select the interface mode. In MII mode (RCR[MII_MODE] = 1 and RCR[RMII_MODE] = 0), there are 18 signals defined by the IEEE 802.3 standard and supported by the EMAC. *Table 804* shows these signals.

Table 804. MII mode

Signal description	EMAC pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[3:0]
Transmit Error	FEC_TXER
Collision	FEC_COL
Carrier Sense	FEC_CRS
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[3:0]
Receive Error	FEC_RXER
Management Data Clock	FEC_MDC
Management Data Input/Output	FEC_MDIO

In RMII mode (RCR[MII_MODE] = 1 and RCR[RMII_MODE] = 1), the EMAC supports 8 external signals. These signals are shown in *Table 805* below.

Table 805. RMII mode configuration

Signal description	EMAC pin
Reference Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[1:0]
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[1:0]
Receive Error	FEC_RXER

The 7-wire serial mode interface (RCR[MII_MODE] = 1 and RCR[RMII_MODE] = 0 or 1) is generally referred to as AMD mode. *Table 806* shows the 7-wire mode connections to the external transceiver.

Table 806. 7-Wire mode configuration

Signal description	EMAC pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[0]
Collision	FEC_COL

Table 806. 7-Wire mode configuration(Continued)

Signal description	EMAC pin
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[0]

48.5.9 FEC frame transmission

The Ethernet transmitter is designed to work with almost no intervention from software. After ECR[ETHER_EN] is set and data appears in the transmit FIFO, the Ethernet MAC can transmit onto the network. The Ethernet controller transmits bytes least significant bit (lsb) first.

When the transmit FIFO fills to the watermark (defined by TFWR), MAC transmit logic asserts FEC_TXEN and starts transmitting the (PA) sequence, the start frame delimiter (SFD), and then the frame information from the FIFO. However, the controller defers the transmission if the network is busy (FEC_CRS is asserted). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense stays inactive for 60 bit times. If so, transmission begins after waiting an additional 36 bit times (96 bit times after carrier sense originally became inactive). See *Section 48.5.19.1: Transmission errors*, for more details.

If a collision occurs during transmission of the frame (half duplex mode), the Ethernet controller follows the specified backoff procedures and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data is transmitted, FCS (frame check sequence) or 32-bit cyclic redundancy check (CRC) bytes are appended if the TC bit is set in the transmit frame control word. If the ABC bit is set in the transmit frame control word, a bad CRC is appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Transmit logic automatically pads short frames (if the TC bit in the transmit buffer descriptor for the end of frame buffer is set).

Settings in the EIMR determine interrupts generated to the buffer (TXB) and frame (TXF).

The transmit error interrupts are HBERR, BABT, LC, RL, and UN. If the transmit frame length exceeds MAX_FL bytes, BABT interrupt is asserted. However, the entire frame is transmitted (no truncation).

To pause transmission, set TCR[GTS] (graceful transmit stop). The FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame finishes or terminates with a collision. After the transmitter has stopped, the GRA (graceful stop complete) interrupt is asserted. If TCR[GTS] is cleared, the FEC resumes transmission with the next frame.

48.5.9.1 Duplicate frame transmission

The FEC fetches transmit buffer descriptors (TxBDs) and the corresponding transmit data continuously until the transmit FIFO is full. It does not determine whether the TxBD to be

fetched is already being processed internally (as a result of a wrap). As the FEC nears the end of the transmission of one frame, it begins to DMA the data for the next frame. To remain one BD ahead of the DMA, it also fetches the TxBD for the next frame. It is possible that the FEC fetches from memory a BD that has already been processed but not yet written back (it is read a second time with the R bit remains set). In this case, the data is fetched and transmitted again.

Using at least three TxBDs fixes this problem for large frames, but not for small frames. To ensure correct operation for large or small frames, one of the following must be true:

- The FEC software driver ensures that there is always at least one TxBD with the ready bit cleared.
- Every frame uses more than one TxBD and every TxBD but the last is written back immediately after the data is fetched.
- The FEC software driver ensures a minimum frame size, *n*. The minimum number of TxBDs is then (Tx FIFO Size ÷ (*n* + 4)) rounded up to the nearest integer (though the result cannot be less than three). The default Tx FIFO size is 192 bytes; this size is programmable.

48.5.10 FEC frame reception

The FEC receiver works with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking. The Ethernet controller receives serial data LSB first.

When the driver enables the FEC receiver by setting ECR[ETHER_EN], it immediately starts processing receive frames. When FEC_RXDV is asserted, the receiver first checks for a valid PA/SFD header. If the PA/SFD is valid, it is stripped and the receiver processes the frame. If a valid PA/SFD is not found, the frame is ignored.

In serial mode, the first 16 bit times of RX_D0 following assertion of FEC_RXDV are ignored. Following the first 16 bit times, the data sequence is checked for alternating 1/0s. If a 11 or 00 data sequence is detected during bit times 17 to 21, the remainder of the frame is ignored. After bit time 21, the data sequence is monitored for a valid SFD (11). If a 00 is detected, the frame is rejected. When a 11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes may occur, but if a 00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame are received, the FEC performs address recognition on the frame.

After a collision window (64 bytes) of data is received and if address recognition has not rejected the frame, the receive FIFO signals the frame is accepted and may be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to reject the frame. Therefore, no collision fragments are presented to you except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and after the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, CR, OV, and TR status bits, and the frame length. See *Section 48.5.19.2: Reception errors*, for more details.

Receive buffer (RXB) and frame interrupts (RFINT) may be generated if enabled by the EIMR register. A receive error interrupt is a babbling receiver error (BABR). Receive frames are not truncated if they exceed the max frame length (MAX_FL); however, the BABR

interrupt occurs and the LG bit in the receive buffer descriptor (RxBD) is set. See *Section 48.5.3.2: Ethernet Receive Buffer Descriptor (RxBD)*, for more details.

When the receive frame is complete, the FEC sets the L-bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E-bit. The Ethernet controller next generates a maskable interrupt (RFINT bit in EIR, maskable by RFIEN bit in EIMR), indicating that a frame is received and is in memory. The Ethernet controller then waits for a new frame.

48.5.11 Ethernet address recognition

The FEC filters the received frames based on destination address (DA) type — individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the destination address field. A flowchart for address recognition on received frames appears in the figures below.

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in *Figure 785* illustrates the address recognition decisions made by the receive block, while *Figure 786* illustrates the decisions made by the microcontroller.

If the DA is a broadcast address and broadcast reject (RCR[BC_REJ]) is cleared, then the frame is accepted unconditionally, as shown in *Figure 785*. Otherwise, if the DA is not a broadcast address, then the microcontroller runs the address recognition subroutine, as shown in *Figure 786*.

If the DA is a group (multicast) address and flow control is disabled, then the microcontroller performs a group hash table lookup using the 64-entry hash table programmed in GAUR and GALR. If a hash match occurs, the receiver accepts the frame.

If flow control is enabled, the microcontroller does an exact address match check between the DA and the designated PAUSE DA (01:80:C2:00:00:01). If the receive block determines the received frame is a valid PAUSE frame, the frame is rejected. The receiver detects a PAUSE frame with the DA field set to the designated PAUSE DA or the unicast physical address.

If the DA is the individual (unicast) address, the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that you program in the PALR and PAUR registers. If an exact match occurs, the frame is accepted; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers, IAUR and IALR. In the case of an individual hash match, the frame is accepted. Again, the receiver accepts or rejects the frame based on PAUSE frame detection, shown in *Figure 785*.

If neither a hash match (group or individual) nor an exact match (group or individual) occur, and if promiscuous mode is enabled (RCR[PROM] set), the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

Similarly, if the DA is a broadcast address, broadcast reject (RCR[BC_REJ]) is asserted, and promiscuous mode is enabled, the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

In general, when a frame is rejected, it is flushed from the FIFO.

Figure 785. Ethernet address recognition—receive block decisions

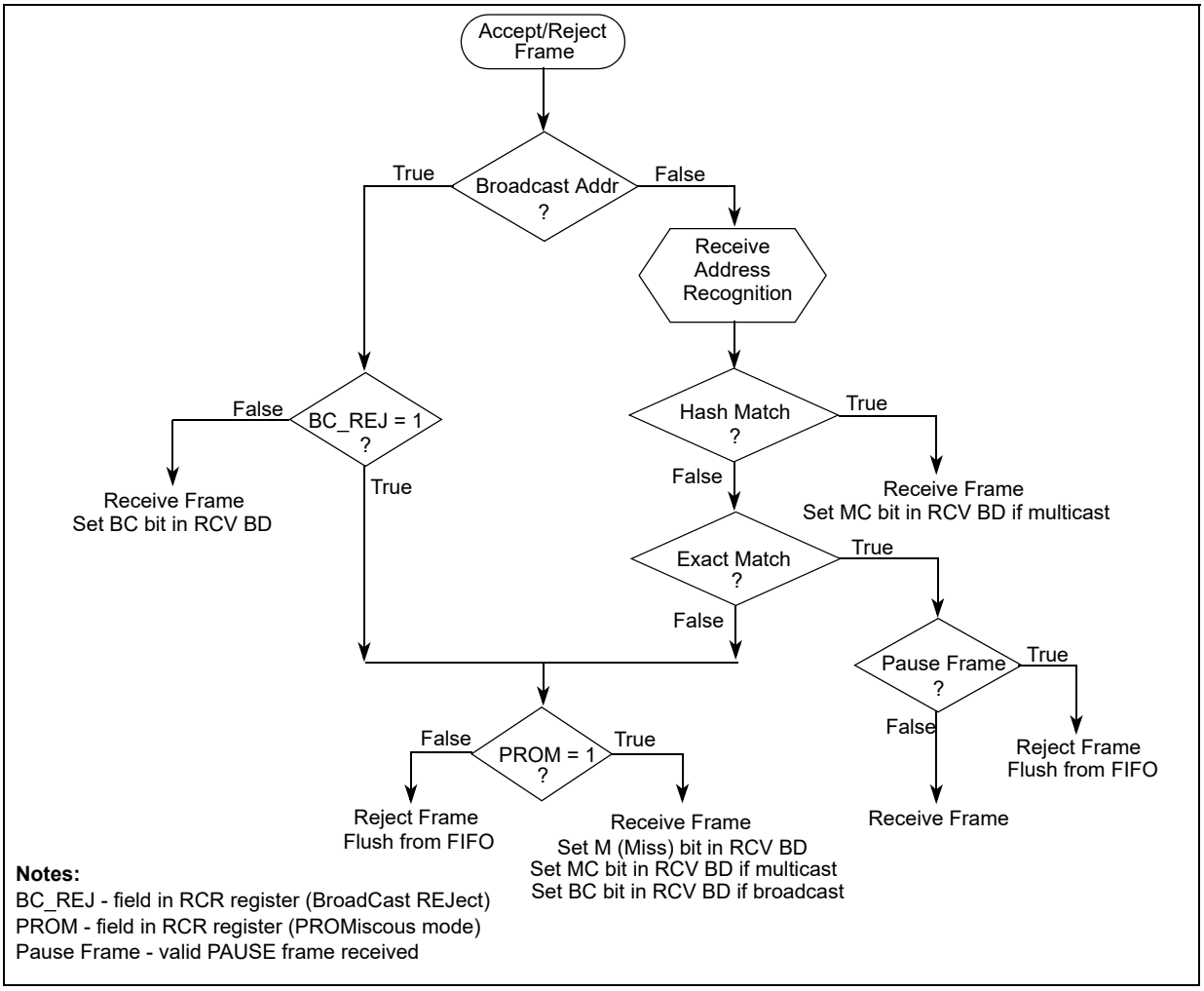
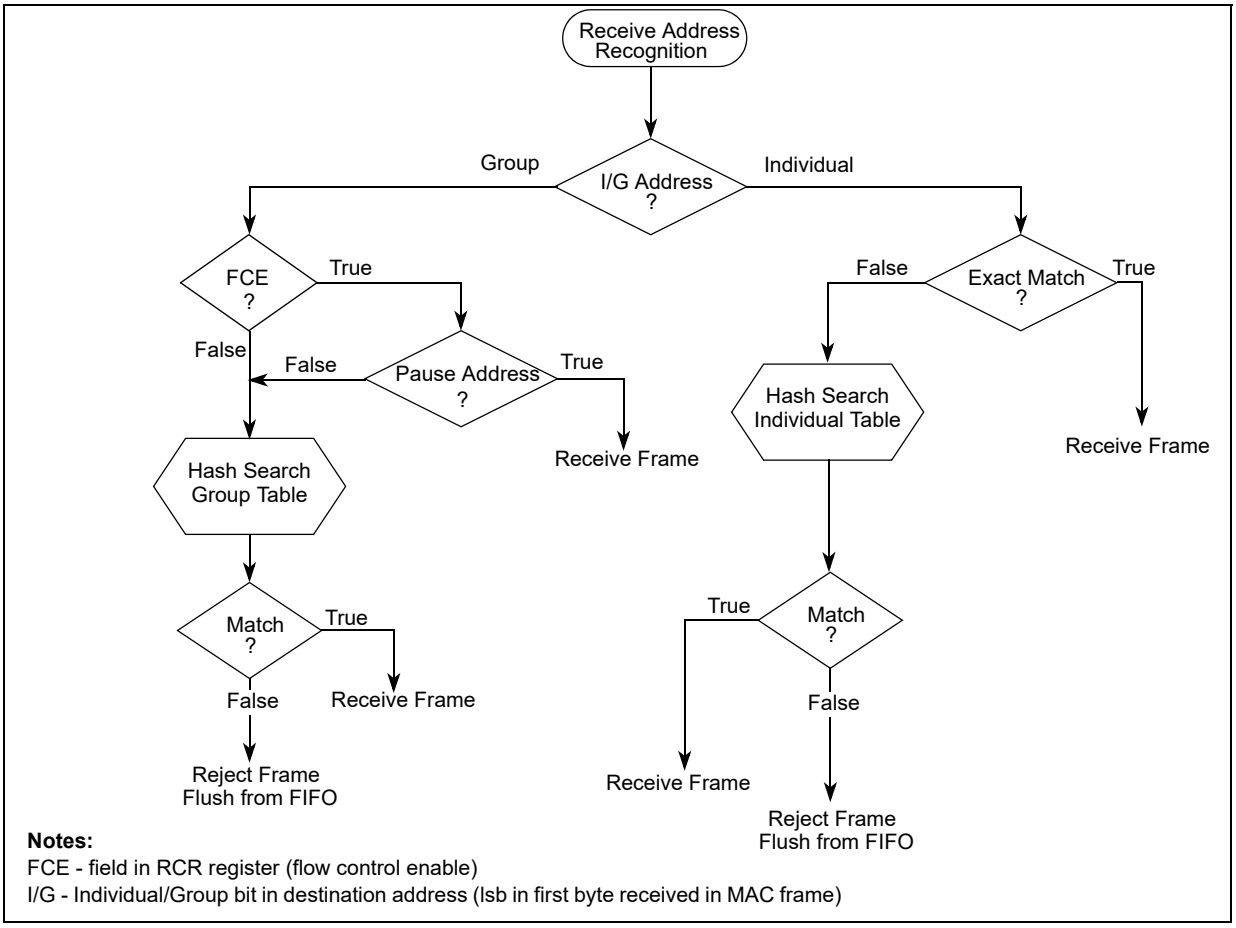


Figure 786. Ethernet address recognition—microcode decisions



48.5.12 Hash algorithm

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits stored in GAUR, GALR (group address hash match), or IAUR, IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63. The msb of the CRC result selects GAUR (msb = 1) or GALR (msb = 0). The five least significant bits of the hash result select the bit within the selected register. If the CRC generator selects a bit set in the hash table, the frame is accepted; otherwise, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

Equation 32

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

Table 807 contains example destination addresses and corresponding hash values.

Table 807. Destination address to 6-bit hash

48-bit DA	6-bit hash (in hex)	Hash decimal value
65FF_FFFF_FFFF	0h	0
55FF_FFFF_FFFF	1h	1
15FF_FFFF_FFFF	2h	2
35FF_FFFF_FFFF	3h	3
B5FF_FFFF_FFFF	4h	4
95FF_FFFF_FFFF	5h	5
D5FF_FFFF_FFFF	6h	6
F5FF_FFFF_FFFF	7h	7
DBFF_FFFF_FFFF	8h	8
FBFF_FFFF_FFFF	9h	9
BBFF_FFFF_FFFF	Ah	10
8BFF_FFFF_FFFF	Bh	11
0BFF_FFFF_FFFF	Ch	12
3BFF_FFFF_FFFF	Dh	13
7BFF_FFFF_FFFF	Eh	14
5BFF_FFFF_FFFF	Fh	15
27FF_FFFF_FFFF	10h	16
07FF_FFFF_FFFF	11h	17
57FF_FFFF_FFFF	12h	18
77FF_FFFF_FFFF	13h	19
F7FF_FFFF_FFFF	14h	20
C7FF_FFFF_FFFF	15h	21
97FF_FFFF_FFFF	16h	22
A7FF_FFFF_FFFF	17h	23
99FF_FFFF_FFFF	18h	24
B9FF_FFFF_FFFF	19h	25
F9FF_FFFF_FFFF	1Ah	26
C9FF_FFFF_FFFF	1Bh	27
59FF_FFFF_FFFF	1Ch	28

Table 807. Destination address to 6-bit hash(Continued)

48-bit DA	6-bit hash (in hex)	Hash decimal value
79FF_FFFF_FFFF	1Dh	29
29FF_FFFF_FFFF	1Eh	30
19FF_FFFF_FFFF	1Fh	31
D1FF_FFFF_FFFF	20h	32
F1FF_FFFF_FFFF	21h	33
B1FF_FFFF_FFFF	22h	34
91FF_FFFF_FFFF	23h	35
11FF_FFFF_FFFF	24h	36
31FF_FFFF_FFFF	25h	37
71FF_FFFF_FFFF	26h	38
51FF_FFFF_FFFF	27h	39
7FFF_FFFF_FFFF	28h	40
4FFF_FFFF_FFFF	29h	41
1FFF_FFFF_FFFF	2Ah	42
3FFF_FFFF_FFFF	2Bh	43
BFFF_FFFF_FFFF	2Ch	44
9FFF_FFFF_FFFF	2Dh	45
DFFF_FFFF_FFFF	2Eh	46
EFFF_FFFF_FFFF	2Fh	47
93FF_FFFF_FFFF	30h	48
B3FF_FFFF_FFFF	31h	49
F3FF_FFFF_FFFF	32h	50
D3FF_FFFF_FFFF	33h	51
53FF_FFFF_FFFF	34h	52
73FF_FFFF_FFFF	35h	53
23FF_FFFF_FFFF	36h	54
13FF_FFFF_FFFF	37h	55
3DFF_FFFF_FFFF	38h	56
0DFF_FFFF_FFFF	39h	57
5DFF_FFFF_FFFF	3Ah	58
7DFF_FFFF_FFFF	3Bh	59
FDFF_FFFF_FFFF	3Ch	60
DDFF_FFFF_FFFF	3Dh	61

Table 807. Destination address to 6-bit hash(Continued)

48-bit DA	6-bit hash (in hex)	Hash decimal value
9DFF_FFFF_FFFF	3Eh	62
BDFF_FFFF_FFFF	3Fh	63

48.5.13 Full duplex flow control

Full-duplex flow control allows you to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, MAC data frame transmission stops for a given pause duration.

To enable PAUSE frame detection, the FEC must operate in full-duplex mode (TCR[FDEN] set) with flow control (RCR[FCE] set). The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in *Table 808*. In addition, the receive status associated with the frame should indicate that the frame is valid.

Table 808. PAUSE frame field specification

48-bit destination address	0180_C200_0001h or Physical Address
48-bit source address	Any
16-bit type	8808h
16-bit opcode	0001h
16-bit PAUSE duration	0000h – FFFFh

The receiver and microcontroller modules perform PAUSE frame detection. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, TCR[GTS] is set by the FEC internally. When transmission has paused, the EIR[GRA] interrupt is asserted and the pause timer begins to increment. The pause timer uses the transmit backoff timer hardware for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments once every slot time, until OPD[PAUSE_DUR] slot times have expired. On OPD[PAUSE_DUR] expiration, TCR[GTS] is cleared allowing MAC data frame transmission to resume. The receive flow control pause status bit (TCR[RFC_PAUSE]) is set while the transmitter pauses due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and you must set flow control pause (TCR[TFC_PAUSE]). After TCR[TFC_PAUSE] is set, the transmitter sets TCR[GTS] internally. When the transmission of data frames stops, the EIR[GRA] (graceful stop complete) interrupt asserts and the pause frame is transmitted. TCR[TFC_PAUSE,GTS] are then cleared internally.

You must specify the desired pause duration in the OPD register.

When the transmitter pauses due to receiver/microcontroller pause frame detection, TCR[TFC_PAUSE] may remain set and cause the transmission of a single pause frame. In this case, the EIR[GRA] interrupt is not asserted.

48.5.14 Inter-Packet Gap (IPG) time

The minimum inter-packet gap time for back-to-back transmission is 96 bit times. After completing a transmission or after the backoff algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96 bit time IPG counter. Frame transmission may begin 96 bit times after carrier sense is negated if it stays negated for at least 60 bit times. If carrier sense asserts during the last 36 bit times, it is ignored and a collision occurs.

The receiver accepts back-to-back frames with a minimum spacing of at least 28 bit times. If an inter-packet gap between receive frames is less than 28 bit times, the receiver may discard the following frame.

48.5.15 Collision managing

If a collision occurs during frame transmission, the Ethernet controller continues the transmission for at least 32 bit times, transmitting a JAM pattern consisting of 32 ones. If the collision occurs during the sequence, a JAM pattern is sent after the end of the preamble sequence.

If a collision occurs within 512 bit times (one slot time), the retry process is initiated. The transmitter waits a random number of slot times. If a collision occurs after 512 bit times, then no retransmission is performed and the end of frame buffer is closed with a Late Collision (LC) error indication.

48.5.16 MII internal and external loopback

Internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Internal and external loopback are configured using combinations of the RCR[LOOP, DRT] and TCR[FDEN] bits.

Set FDEN for internal and external loopback.

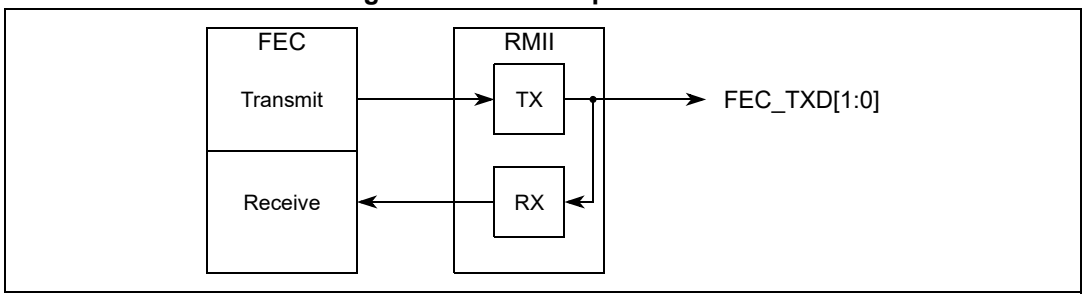
For internal loopback, set RCR[LOOP] and clear RCR[DRT]. FEC_TXEN and FEC_TXER do not assert during internal loopback. During internal loopback, the transmit/receive data rate is higher than in normal operation because the transmit and receive blocks use the internal bus clock instead of the clocks from the external transceiver. This causes an increase in the required system bus bandwidth for transmit and receive data being DMA'd to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underruns and receive FIFO overflows.

For external loopback, clear RCR[LOOP] and RCR[DRT], and configure the external transceiver for loopback.

48.5.17 RMII loopback

The FEC also supports RMII loopback. In this mode, transmit data is sent to the RMII receive logic and out the FEC_TXD[1:0] pins. To enable RMII loopback mode, set RCR[RMII_MODE, RMII_LOOP] and TCR[FDEN] and clear RCR[RMII_ECHO, LOOP].

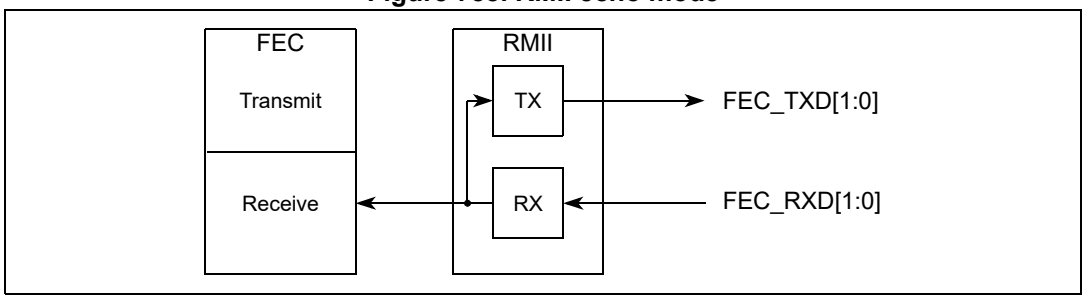
Figure 787. RMII loopback mode



48.5.18 RMII echo

The Ethernet controller also supports RMII echo mode, which transmits data on FEC_TXD[1:0] as it is received on FEC_RXD[1:0]. To enable RMII echo mode, set RCR[RMII_MODE, RMII_ECHO] bits and clear RCR[RMII_LOOP, LOOP].

Figure 788. RMII echo mode



48.5.19 Ethernet error-managing procedure

The Ethernet controller reports frame reception and transmission error conditions using the MIB block counters, the FEC RxBDs, and the EIR register.

48.5.19.1 Transmission errors

48.5.19.1.1 Transmitter underrun

If this error occurs, the FEC sends 32 bits that ensure a CRC error and stops transmitting. All remaining buffers for that frame are then flushed and closed, and EIR[UN] is set. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame. The UN interrupt is asserted if enabled in the EIMR register.

48.5.19.1.2 Retransmission attempts limit expired

When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and EIR[RL] is set. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame. The RL interrupt is asserted if enabled in the EIMR register.

48.5.19.1.3 Late collision

When a collision occurs after the slot time (512 bits starting at the Preamble), the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and

EIR[LC] is set. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame. The LC interrupt is asserted if enabled in the EIMR register.

48.5.19.1.4 Heartbeat

Some transceivers have a self-test feature called heartbeat or signal quality error. To signify a good self-test, the transceiver indicates a collision to the FEC within four microseconds after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver continues to function properly. This is the heartbeat condition.

If TCR[HBC] is set and the heartbeat condition is not detected by the **FEC** after a frame transmission, a heartbeat error occurs. When this error occurs, the FEC closes the buffer, sets EIR[HB], and generates the HBERR interrupt if it is enabled.

48.5.19.2 Reception errors

48.5.19.2.1 Overrun error

If the receive block has data to put into the receive FIFO and the receive FIFO is full, FEC sets RxBD[OV]. All subsequent data in the frame is discarded and subsequent frames may also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit set. The driver must discard this frame.

48.5.19.2.2 Non-octet error (dribbling bits)

The Ethernet controller manages up to seven dribbling bits when the receive frame terminates past an non-octet aligned boundary. Dribbling bits are not used in the CRC calculation. If there is a CRC error, the frame non-octet aligned (NO) error is reported in the RxBD. If there is no CRC error, no error is reported.

48.5.19.2.3 CRC error

When a CRC error occurs with no dribble bits, FEC closes the buffer and sets RxBD[CR]. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

48.5.19.2.4 Frame length violation

When the receive frame length exceeds MAX_FL bytes the BABR interrupt is generated, and RxBD[LG] is set. The frame is not truncated unless the frame length exceeds 2047 bytes.

48.5.19.2.5 Truncation

When the receive frame length exceeds 2047 bytes, frame is truncated and RxBD[TR] is set.

49 SENT Receiver (SRX)

49.1 Introduction

The Single Edge Nibble Transmission (SENT) Receiver (SRX) module is a multi-channel receiver for receiving serial data frames which are being transmitted by a sensor implementing the SENT encoding scheme and present them to the CPU for further processing.

This module is based on the SAE J2716 information report titled “SENT - Single Edge Nibble Transmission for Automotive Applications” and released on January 27, 2010 , Apr2007 and Feb2008. As per this standard, the SENT protocol is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10-bit A/Ds and PWM and as a simpler low-cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

The SENT encoding scheme is a unidirectional communications scheme from the sensor / transmitting device to the controller /receiving device which does not include a coordination signal from the controller/receiving device. The sensor signal is transmitted as a series of pulses with data encoded as falling to falling edge periods.

49.1.1 Features

The SENT receiver module has the following features:

- The number of SENT channels supported are device-specific. See *Section 6.8.5: SENT Receiver (SRX) configuration* for details.
- Unified message read logic to transfer all messages received to system memory via interrupts or DMA
  - Programmable option per channel to support reading of messages via DMA or Interrupt
  - Internal round robin arbitration sequencer to loop across channels to read messages from each channel leaving CPU/DMA transparent to this process
  - All received messages are stored excluding the synchronization pulse (for Fast messages) and identifiers (for Slow Serial Messages)
  - Separate buffers for storing Fast and Slow Serial Messages
  - Separate DMA and Interrupts generated for Fast and Slow Serial Messages
- Supports compensation for variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift and jitter in the Transmitter and Receiver clocks per channel
  - Calibration pulse correction factor per channel is available for use by software
- Implements for each channel all receiver diagnostics as specified by the SAE Specifications
- Internal 4-bit CRC calculator that can be configured to compute CRC in both Legacy and Recommended method (as given in SAE SENT Specification) for both Fast and Short Serial Message per channel
- Internal 6-bit CRC calculator for Enhanced Serial Messages per channel
- Supports all bit rates by having per channel configurable Rx clock periods from 3 µs to 90 µs
- Support for configurable number of data nibbles per channel
- Option to disable individual channel via programmable control bit
- Time stamping on start of each valid message (i.e. without errors) across all channels. This time stamp is appended to each message for synchronization of messages and for application to understand how much time has elapsed between successive sensor readings
- Support for pause pulse per channel. This can be enabled by programming a bit in the control register for that channel
- Input Programmable Filter on each channel to filter any glitches on input. This filter duration is programmable by the user software.
- 32-bit register interface for programming the module's registers and reading the received messages
- Works on two clocks, one for accurate receive operations (high frequency receiver clock or protocol clock) and other for message read logic (system bus clock) and the module can work irrespective of the frequency relationship between the two clocks. See *Section 49.5: Clocks and resets on page 1372* for more details on the clocks.

49.1.2 Modes of operation

49.1.2.1 Debug mode

In debug mode, message reception will be halted and all existing status and messages will be retained. DBG_FRZ bit is required to be set for this mode. On debug exit, a calibration length error (CAL_LEN_ERR) can be reported and the channels will start receiving messages from a valid calibration pulse.

49.1.2.2 Low power modes

SENT Receiver behavior is affected by chip-specific low power modes (for example, STOP or WAIT modes). These low power modes should either gate-off both the system bus clock and the protocol clocks or reduce their frequencies, depending on the application.

On exiting from these low power modes, calibration length error (CAL_LEN_ERR) can get set in channel status registers as the SENT sensor (transmitter) could be in the middle of a message transmission at that point. Thus, software should clear all error status bits after exiting the low power modes.

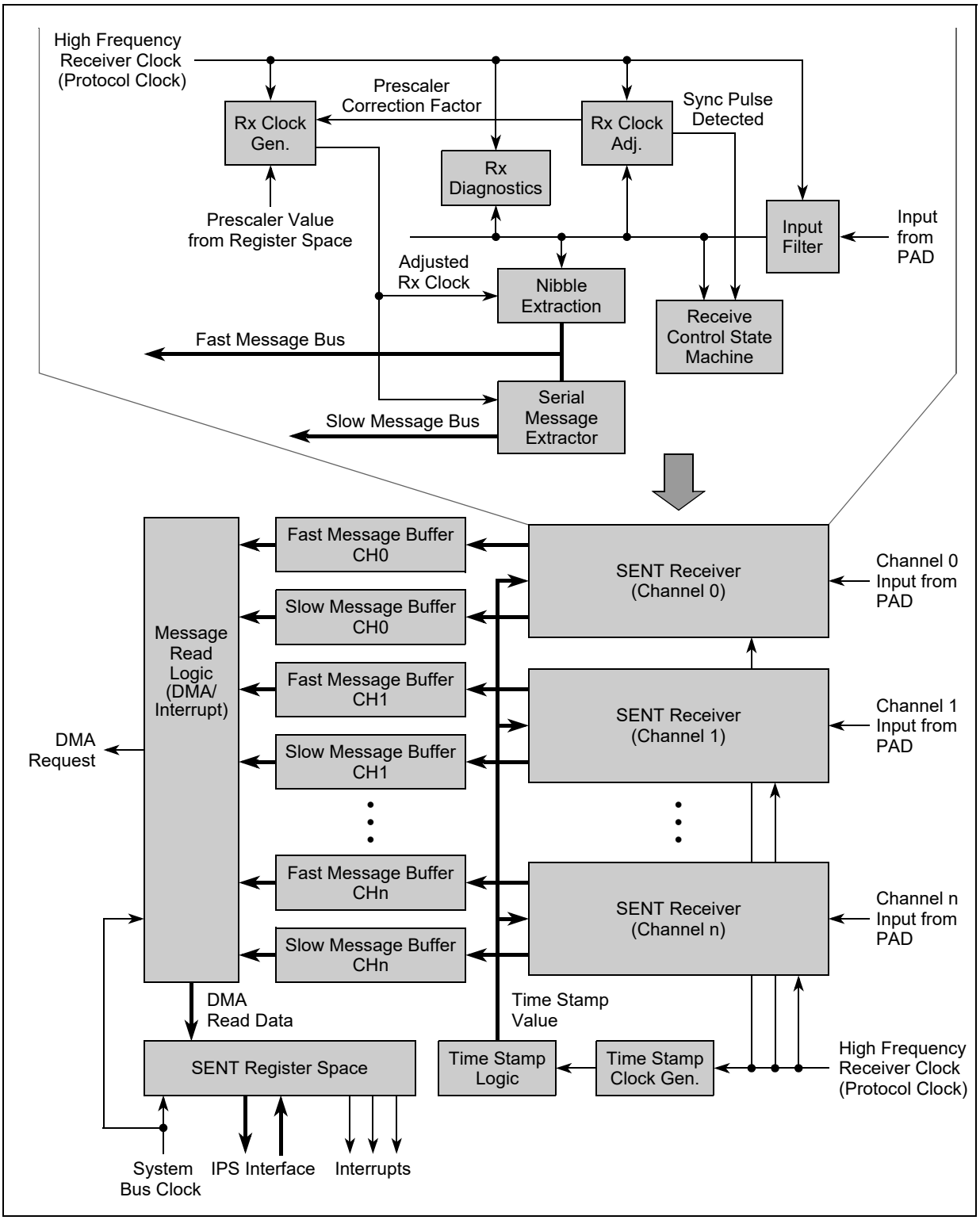
Modes where both system bus clock and protocol clock are gated: In this case, both message reception and interrupt / DMA request generation will halt. The internal receiver channels will freeze (since clock is not there) and will not receive any messages. Messages which are completely received before entering the low power mode can be read after the low power mode exit.

Modes where frequency of system bus clock is changed: For reduced power consumption the system bus clock can be reduced but not below a minimum value (See *Section 49.5.2: System bus clock requirements on page 1373*) to avoid overflow of received messages. Protocol clock should not go below the minimum value (See *Section 49.5.3: High frequency receiver clock (protocol clock) requirements on page 1374*) for accurate receive operations.

Note: Gating of clocks in the low power mode is chip-specific, see the chapter that refers how the modules behave in low power modes.

49.1.3 Block diagram

Figure 789. SENT block diagram



49.1.4 Design overview

The SENT Receiver Module is a unified receiver module comprising of a number of receiver channels and a unified DMA and Register Space for accessing the messages from each channel. *Figure 789* shows the top level block diagram for the SENT Receiver module.

Each channel of the SENT Receiver is composed of a Clock Generation and Adjustment block to generate the receiver clock and adjust it for frequency variation in the transmitter clock. This block also detects the synchronization/calibration pulse which is present in every SENT message.

The SENT Receiver State Machine controls the sequencing of decoding the messages and combining the information as nibbles. In parallel a diagnostic block is running that monitors the incoming messages for any errors and also verifies the CRC of the received messages. The fast and serial message extractors collate the message nibbles and add the channel number and time stamp information to the messages.A free running time stamp counter is used to add the time stamp value to each message received. The value of this counter is input to each channel and the channel logic appends the time stamp to each message. These messages are then formatted to be stored in the corresponding message read buffer. Only the messages that pass the diagnostics checks are forwarded to be stored in the buffers.

The SENT Receiver provides two mechanisms for reading the received messages, which are via DMA transfers and via Interrupts. Based on the control bit settings in the control registers, a DMA request or Interrupt is asserted to signal the DMA controller or CPU that messages are available to be read.

The free running time stamp counter, register block and DMA interface are common for all channels configured. The register space contains all the necessary registers for all channels. The register space is 32-bit aligned and accessible via 8-/16-/32-bit accesses. The registers for all un-configured channels are marked as reserved.

49.1.4.1 Functional overview

After reset, the SENT module comes up in the disabled state. All channel enable bits and the global enable bit are set to 0 and the user software must program the SENT module parameters correctly before enabling it. The user software must correctly configure all parameters to avoid any underflow or overflow in the message buffers.

Once enabled, the Receiver Clock Generation and Adjustment block detects the synchronization pulse. Until then, the state machine blocks all subsequent operations of the receiver. Once the synchronization pulse is detected, the receiver clock tick period is compensated for variation in transmitter clock tick period and the state machine then waits for the reception of data nibbles. The adjusted receiver clock is used to sample the data nibbles from the incoming serial message. The state machine waits until the status and communication nibble and configured number of data nibbles are received. The state machine then waits for the pause pulse (if configured to receive a pause pulse). The fast and serial message extractor block provides the state machine with the information on how many nibbles have been received. All diagnostics checks are computed in parallel. Once all required data nibbles are received, the CRC is checked, time stamp and channel number is added to the message. The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer along with the message. This final message is stored in the message buffer as described in the

subsequent sections. Any error detected during the receive process causes the status bits in the corresponding channel's status register to be set.

The above process repeats across each enabled channel in parallel and keeps pushing error free messages into the buffers on high frequency receiver clock. The module's message read logic synchronizes all these messages on the bus clock and pops out these messages on any DMA or CPU read access. Any DMA underflow errors are stored in the global status register while the overflow errors are stored in the corresponding channel's status register. In case of an overflow in Fast Messages, the newer messages received will overwrite the previous message.

For detailed description on the complete functionality of each block, see *Section 49.4: Functional description on page 1361*.

49.2 External signal description

This section lists all inputs to the SENT block.

49.2.1 SENT_RX [(CH-1):0]

These pins serve as the serial data input for each of the SENT channel. There is one bit per channel supported by the device. The SENT sensor input pads on the device using this module should have pull-up enabled to ensure a 1'b1 is seen at the module input when SENT sensor is not driving any data.

Note: The value of the CH is device-specific. See Section 6.8.5: SENT Receiver (SRX) configuration for details.

49.3 Memory map and register definition

This section provides a detailed description of all registers accessible in the SENT block.

49.3.1 Memory map

Table 809 gives an overview on all SENT registers. The memory map comprises of 32-bit aligned registers which can be accessed via 8-bit, 16-bit, or 32-bit accesses. Write access to reserved locations will generate transfer error. Read access to reserved locations will also generate transfer error and the read data bus will show all 0s. The Memory Map and complete module is in Big Endian Format.

The module will not check for correctness of programmed values in Register Space and software must ensure that correct values are being written.

Note: Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level. The number of registers are device specific. See the Device Configuration chapter for details.

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

Table 809. SENT memory map

Offset	Register Name	Access	Reset	Location
0x0000	GBL_CTRL— Global Control Register	RW	0x0000_0010	*49.3.2.1/1330*
0x0004	CHNL_EN—Channel Enable Register	RW	0x0000_0000	*49.3.2.2/1332*
0x0008	GBL_STATUS—Global Status Register	W1C	0x0000_0000	*49.3.2.3/1333*
0x000C	FMSG_RDY—Fast Message Ready Register	RW(1)	0x0000_0000	*49.3.2.4/1334*
0x0010	SMSG_RDY—Slow Serial Message Ready Register	RW(1)	0x0000_0000	*49.3.2.5/1336*
0x0014	Reserved			
0x0018	DATA_CTRL1— Data Control Register 1	RW	0x1111_1111	*49.3.2.6/1338*
0x001C	DATA_CTRL2— Data Control Register 2	RW	0x1111_1111	*49.3.2.7/1340*
0x0020	Reserved			
0x0024	Reserved			
0x0028	FDMA_CTRL—Fast Message DMA Control Register	RW	0x0000_0000	*49.3.2.8/1341*
0x002C	SDMA_CTRL—Slow Serial Message DMA Control Register	RW	0x0000_0000	*49.3.2.9/1342*
0x0030	Reserved			
0x0034	FRDY_IE—Fast Message Ready Interrupt Control Register	RW	0x0000_0000	*49.3.2.10/1343*
0x0038	SRDY_IE—Slow Serial Message Ready Interrupt Control Register	RW	0x0000_0000	*49.3.2.11/1344*
0x003C	Reserved			
0x0040	DMA_FMSG_DATA—DMA Fast Message Read Register	R	0x0000_0000	*49.3.2.12/1345*
0x0044	DMA_FMSG_CRC—DMA Fast Message CRC Read Register	R	0x0000_0000	*49.3.2.13/1347*
0x0048	DMA_FMSG_TS—DMA Fast Message Time Stamp Read Register	R	0x0000_0000	*49.3.2.14/1347*
0x004C	Reserved			
0x0050	DMA_SMSG_BIT3—DMA Serial Message Read Register (Bit 3)	R	0x0000_0000	*49.3.2.15/1348*
0x0054	DMA_SMSG_BIT2—DMA Serial Message Read Register (Bit 2)	R	0x0000_0000	*49.3.2.16/1350*
0x0058	DMA_SMSG_TS—DMA Serial Message Time Stamp Read Register	R	0x0000_0000	*49.3.2.17/1351*
0x005C	Reserved			
0x0060 + 0x10 × n	CHn_CLK_CTRL—Channel ‘n’ Clock Control Register (n = 0 to (CH-1))	RW	0x0000_8000	*49.3.2.18/1352*
0x0064 + 0x10 × n	CHn_STATUS—Channel ‘n’ Status Register (n = 0 to (CH-1))	W1C	0x0000_0000	*49.3.2.19/1353*

Table 809. SENT memory map(Continued)

Offset	Register Name	Access	Reset	Location
0x0068 + 0x10 × n	CHn_CONFIG—Channel ‘n’ Configuration Register (n = 0 to (CH-1))	RW	0x0000_0104	*49.3.2.20/1356*
0x006C + 0x10 × n	Reserved			
0x0160 + 0x18 × n	CHn_FMSG_DATA—Channel ‘n’ Fast Message Register (n = 0 to (CH-1))	R	0x0000_0000	*49.3.2.21/1359*
0x0164 + 0x18 × n	CHn_FMSG_CRC—Channel ‘n’ Fast Message CRC Register (n = 0 to (CH-1))	R	0x0000_0000	*49.3.2.22/1360*
0x0168 + 0x18 × n	CHn_FMSG_TS—Channel ‘n’ Fast Message Time Stamp Register (n = 0 to (CH-1))	R	0x0000_0000	*49.3.2.23/1360*
0x016C + 0x18 × n	CHn_SMSG_BIT3—Channel ‘n’ Serial Message Register (Bit 3) (n = 0 to (CH-1))	R	0x0000_0000	*49.3.2.24/1360*
0x0170 + 0x18 × n	CHn_SMSG_BIT2—Channel ‘n’ Serial Message Register (Bit 2) (n = 0 to (CH-1))	R	0x0000_0000	*49.3.2.25/1360*
0x0174 + 0x18 × n	CHn_SMSG_TS—Channel ‘n’ Serial Message Time Stamp Register (n = 0 to (CH-1))	R	0x0000_0000	*49.3.2.26/1360*
0x02E0	Reserved			

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the *Section 49.3.2.1: Global Control Register (GBL_CTRL)*

49.3.2 Register descriptions

This section describes in address order all the SENT registers and their individual bits.

49.3.2.1 Global Control Register (GBL_CTRL)

This register provides the global control and setting for the SENT Receiver Module. The SENT_EN bit must be programmed after all other modules settings and control bits have been written to.

Address Offset = 0x0000 Access: RW

	0	1	2	3	4	5	6	7
R	TSPRSC							
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R							FMDUIE	SMDUIE
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R				FAST_CLR		DBG_FRZ		SENT_EN
W								
Reset	0	0	0	1	0	0	0	0

Figure 790. Global Control Register (GBL_CTRL)

Table 810. GBL_CTRL field descriptions

Field	Description
0:7 TSPRSC	Time Stamp Prescaler Value. Value in number of clock cycles of high frequency receiver clock set in order to obtain a clock frequency for the time stamp counter block. 0 – Prescaler Bypassed. Time Stamp Clock = High Frequency Receiver Clock 1 to 255 – Value set by user software to obtain a clock period of at least 1*us* for the time stamp counter. If the high frequency clock input is 75 MHz, then the value set in this field would be 0x4B (or 75) to obtain a 1µs clock. **Note:** Since the time stamp clock is common for all channels, the timestamp clock frequency must be at least twice the fastest receiver channel in the application. This is to ensure that the time stamp clock is fast enough to show change in timestamp from message to message.
8:15	Reserved. Read returns zero
16:21	Reserved. Read returns zero
22 FMDUIE	Fast Message DMA Underflow Interrupt Enable: Enables interrupt assertion when there is an underflow condition on Fast Message DMA registers. Default value is 0. 0 – Interrupt is disabled 1 – Interrupt is enabled

Table 810. GBL_CTRL field descriptions(Continued)

Field	Description
23 SMDUIE	Slow Serial Message DMA Underflow Interrupt Enable: Enables interrupt assertion when there is an underflow condition on Slow Message DMA registers. Default value is 0. 0 – Interrupt is disabled 1 – Interrupt is enabled
24	Reserved. Read returns zero
25	Reserved. Read returns zero
26	Reserved. Read returns zero
27 FAST_CLR	Fast Clearing Enable bit. Enables clearing of ready status (for both fast and slow) bit automatically when corresponding message is read. Default value is 1. See *Section 49.3.2.4: Fast Message Ready Status Register (FMSG_RDY)* and *Section 49.3.2.5: Slow Serial Message Ready Status Register (SMSG_RDY)* for details on the ready status bits. 0 – Fast Clearing is disabled. Bits will be cleared by writing 1 1 – Fast Clearing is enabled
28	Reserved. Read returns zero
29 DBG_FRZ	Debug Freeze. This bit will enable the debug mode support. SENT Module will freeze in debug mode if this bit is set. Default is 0. 0 – No effect in debug mode 1 – Freeze in debug mode
30	Reserved. Read returns zero
31 SENT_EN	SENT Receiver Global Enable. This bit enables or disables the complete receiver module irrespective of the setting of individual channel enable bits. User software must program all channel parameters before setting this bit. Default is 0. 0 – Entire module is disabled 1 – Module is enabled

49.3.2.2 Channel Enable Register (CHNL_EN)

This register is used to enable individual SENT Receiver channels in the module. The user software must ensure all channel parameters and control settings have been programmed before writing to this bit. In case these parameters are changed while the channel is enabled, normal operation is not guaranteed.

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. See Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and these bits must be written to 0.

Address Offset: 0x0004 Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	EN_CH15	EN_CH14	EN_CH13	EN_CH12	EN_CH11	EN_CH10	EN_CH9	EN_CH8
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	EN_CH7	EN_CH6	EN_CH5	EN_CH4	EN_CH3	EN_CH2	EN_CH1	EN_CH0
W								
Reset	0	0	0	0	0	0	0	0

Figure 791. Channel Enable Register (CHNL_EN)

Table 811. CHNL_EN field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 EN_CHn (n = 15 to 0)	Enable bits for Channels 0 to 15. All channels are disabled after reset. User software must program all channel parameters before setting this bit. Once set, other channel parameters in registers should not be changed until this bit is cleared. Only Interrupt Enable and Status bits can be written to. 0 – Channel is disabled 1 – Channel is enabled. Channel parameter registers are locked for writing

49.3.2.3 Global Status Register (GBL_STATUS)

This register provides the status of various conditions in the module that are common across all channels.

Address Offset = 0x0008 Access: W1C

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R							FMDU	SMDU
W							w1c	w1c
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R								
W								
Reset	0	0	0	0	0	0	0	0

Figure 792. Global Status Register (GBL_STATUS)

Table 812. GBL_STATUS field descriptions

Field	Description
0:21	Reserved. Read returns zero
22 FDMU	Fast Message DMA Underflow: Interrupt Status bit that indicates when there is an underflow in Fast Message buffers in any channel which has DMA enabled. This bit is set when there is no message in the DMA buffers for read and the software still reads the Fast Message DMA register. Default value is 0. 0 – No underflow 1 – Underflow occurred
23 SMDU	Slow Serial Message DMA Underflow: Interrupt Status bit that indicates when there is an Underflow in Slow Serial Message DMA buffers on channel which has DMA enabled. The bit is set when there is no message in the DMA buffers for read and the software still reads the Slow Serial Message registers. Default value is 0. 0 – No underflow 1 – Underflow occurred
24:31	Reserved. Read returns zero

49.3.2.4 Fast Message Ready Status Register (FMSG_RDY)

The bits of this register are asserted when a Fast Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA

(via *Fast Message DMA Control Register (FDMA_CTRL)*), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (via *Section 49.3.2.10: Fast Message Ready Interrupt Control Register (FRDY_IE)*) and the channel is enabled, the message ready status bit in this register will get asserted on Fast Message reception. Software must ensure that if a channel is enabled, either DMA or Interrupt must be enabled for that channel. Proper flow cannot be guaranteed if neither is selected.

These bits can be configured to be cleared via “Write 1 to Clear” or “Fast Clearing Mechanism”. The configuration is done by writing to the FAST_CLR bit in the *Section 49.3.2.1: Global Control Register (GBL_CTRL)*.

Note: The following register figure and table show the maximum possible configuration but the exact number of valid register bits depends on the supported SENT channels on the device. See the Device Configuration chapter for the number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are unadvisable and may cause unexpected behavior.

Address Offset: 0x000C Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	F_RDY15	F_RDY14	F_RDY13	F_RDY12	F_RDY11	F_RDY10	F_RDY9	F_RDY8
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	F_RDY7	F_RDY6	F_RDY5	F_RDY4	F_RDY3	F_RDY2	F_RDY1	F_RDY0
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0

Figure 793. Fast Message Ready Status Register (FMSG_RDY)

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the *Section 49.3.2.1: Global Control Register (GBL_CTRL)*.

Table 813. FMSG_RDY field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 F_RDYn (n = 15 to 0)	Fast Message Data Ready Status Bit. Indicates that a Fast Message is ready to be read by CPU from that specific channel. This bit is asserted only when **DMA is NOT** enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Fast Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). 0 – No Fast Message is available to be read out via Interrupt 1 – Fast Message is available to be read out via Interrupt The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section *Section 49.3.2.1: Global Control Register (GBL_CTRL)* for details on FAST_CLR bit. **Note:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 fast message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. **Note:** When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits.

49.3.2.5 Slow Serial Message Ready Status Register (SMSG_RDY)

The bits of this register are asserted when a Slow Serial Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (via *Section 49.3.2.8: Fast Message DMA Control Register (FDMA_CTRL)* and *Section 49.3.2.9: Slow Serial Message DMA Control Register (SDMA_CTRL)*), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (via *Section 49.3.2.10: Fast Message Ready Interrupt Control Register (FRDY_IE)* and *Section 49.3.2.11: Slow Serial Message Ready Interrupt Enable Register (SRDY_IE)*) and the channel is enabled, the message ready status bit in this register will get asserted on Slow Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Erroneous flow can happen if neither is selected.

These bits can be configured to be cleared via “Write 1 to Clear” or “Fast Clearing Mechanism”. The configuration is done by writing to the FAST_CLR bit in the *Global Status Register (GBL_STATUS)*.

Note: The following register figure and table show the maximum possible configuration but the exact number of valid register bits is dependent on the supported SENT channels on the device. See the Device Configuration chapter for the number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are unadvisable and may cause unexpected behavior.

Address Offset = 0x0010 Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	S_RDY15	S_RDY14	S_RDY13	S_RDY12	S_RDY11	S_RDY10	S_RDY9	S_RDY8
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	S_RDY7	S_RDY6	S_RDY5	S_RDY4	S_RDY3	S_RDY2	S_RDY1	S_RDY0
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0

Figure 794. Slow Serial Message Ready Status Register (SMSG_RDY)

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the *Section 49.3.2.1: Global Control Register (GBL_CTRL)*.

Table 814. SMSG_RDY field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 S_RDYn (n = 15 to 0)	Slow Serial Message Data Ready Status Bit. Indicates that a Slow Serial Message is ready to be read by CPU from that specific channel. This bit is asserted only when **DMA is NOT** enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Serial Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). 0 – No Slow Serial Message is available to be read out via Interrupt 1 – Slow Serial Message is available to be read out via Interrupt The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section *Global Control Register (GBL_CTRL)* for details on FAST_CLR bit. **Note:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 Fast Message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. **Note:** When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits.

49.3.2.6 Data Control Register 1 (DATA_CTRL1)

The Data Control Register 1 (DATA_CTRL1) is described below.

Address Offset = 0x0018 Access: RW

	0	1	2	3	4	5	6	7
R		NIBBCH0				NIBBCH1		
W								
Reset	0	0	0	1	0	0	0	1

	8	9	10	11	12	13	14	15
R		NIBBCH2				NIBBCH3		
W								
Reset	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23
R		NIBBCH4				NIBBCH5		
W								
Reset	0	0	0	1	0	0	0	1

	24	25	26	27	28	29	30	31
R		NIBBCH6				NIBBCH7		
W								
Reset	0	0	0	1	0	0	0	1

Figure 795. Data Control Register 1 (DATA_CTRL1)

Table 815. DATA_CTRL1 field descriptions

Field	Description
0	Reserved for future use. Reads returns zeros.
1:3 NIBBCH0	Number of Data Nibbles supported in Channel 0 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
4	Reserved for future use. Reads returns zeros.
5:7 NIBBCH1	Number of Data Nibbles supported in Channel 1 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
8	Reserved for future use. Reads returns zeros.
9:11 NIBBCH2	Number of Data Nibbles supported in Channel 2 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel

Table 815. DATA_CTRL1 field descriptions(Continued)

Field	Description
12	Reserved for future use. Reads returns zeros.
13:15 NIBBCH3	Number of Data Nibbles supported in Channel 3 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
16	Reserved for future use. Reads returns zeros.
17:19 NICCH4	Number of Data Nibbles supported in Channel 4 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
20	Reserved for future use. Reads returns zeros.
21:23 NIBBCH5	Number of Data Nibbles supported in Channel 5 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
24	Reserved for future use. Reads returns zeros.
25:27 NIBBCH6	Number of Data Nibbles supported in Channel 6 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
28	Reserved for future use. Reads returns zeros.
29:31 NIBBCH7	Number of Data Nibbles supported in Channel 7 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel

49.3.2.7 Data Control Register 2 (DATA_CTRL2)

Address Offset = 0x001C Access: RW

	0	1	2	3	4	5	6	7
R		NIBBCH8				NIBBCH9		
W								
Reset	0	0	0	1	0	0	0	1

	8	9	10	11	12	13	14	15
R		NIBBCH10				NIBBCH11		
W								
Reset	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23
R		NIBBCH12				NIBBCH13		
W								
Reset	0	0	0	1	0	0	0	1

	24	25	26	27	28	29	30	31
R		NIBBCH14				NIBBCH15		
W								
Reset	0	0	0	1	0	0	0	1

Figure 796. Data Control Register 2 (DATA_CTRL2)

Table 816. DATA_CTRL2 field descriptions

Field	Description
0	Reserved for future use. Reads returns zeros.
1:3 NIBBCH8	Number of Data Nibbles supported in Channel 8 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
4	Reserved for future use. Reads returns zeros.
5:7 NIBBCH9	Number of Data Nibbles supported in Channel 9 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
8	Reserved for future use. Reads returns zeros.
9:11 NIBBCH10	Number of Data Nibbles supported in Channel 10 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
12	Reserved for future use. Reads returns zeros.

Table 816. DATA_CTRL2 field descriptions(Continued)

Field	Description
13:15 NIBBCH11	Number of Data Nibbles supported in Channel 11 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
16	Reserved for future use. Reads returns zeros.
17:19 NIBBCH12	Number of Data Nibbles supported in Channel 12 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
20	Reserved for future use. Reads returns zeros.
21:23 NIBBCH13	Number of Data Nibbles supported in Channel 13 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
24	Reserved for future use. Reads returns zeros.
25:27 NIBBCH14	Number of Data Nibbles supported in Channel 14 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel
28	Reserved for future use. Reads returns zeros.
29:31 NIBBCH15	Number of Data Nibbles supported in Channel 15 0 and 7 – Invalid value. Must not be used 1 to 6 – Number of data nibbles to be supported by respective channel

49.3.2.8 Fast Message DMA Control Register (FDMA_CTRL)

Note: The following register figure and table show the maximum possible configuration but the exact number of valid register bits is dependent on the supported SENT channels on the device. See the Device Configuration chapter for the number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are unadvisable and may cause unexpected behavior.

Address Offset: 0x0028 Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	FDMA_EN15	FDMA_EN14	FDMA_EN13	FDMA_EN12	FDMA_EN11	FDMA_EN10	FDMA_EN9	FDMA_EN8
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	FDMA_EN7	FDMA_EN6	FDMA_EN5	FDMA_EN4	FDMA_EN3	FDMA_EN2	FDMA_EN1	FDMA_EN0
W								
Reset	0	0	0	0	0	0	0	0

Figure 797. Fast Message DMA Control Register (FDMA_CTRL)

Table 817. FDMA_CTRL field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 FDMA_ENn (n = 15 to 0)	Enable DMA for Fast Messages on Channels 0 to 15. These bits are writeable when corresponding Fast Message Ready Interrupt Enable bits are set to 0. 0 – DMA for Fast Messages is disabled 1 – DMA for Fast Messages is enabled

49.3.2.9 Slow Serial Message DMA Control Register (SDMA_CTRL)

Note: The following register figure and table show the maximum possible configuration but the exact number of valid register bits is dependent on the supported SENT channels on the device. See the Device Configuration chapter for the number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are unadvisable and may cause unexpected behavior.

Address Offset: 0x002C Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	SDMA_EN15	SDMA_EN14	SDMA_EN13	SDMA_EN12	SDMA_EN11	SDMA_EN10	SDMA_EN9	SDMA_EN8
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	SDMA_EN7	SDMA_EN6	SDMA_EN5	SDMA_EN4	SDMA_EN3	SDMA_EN2	SDMA_EN1	SDMA_EN0
W								
Reset	0	0	0	0	0	0	0	0

Figure 798. Slow Serial Message DMA Control Register (SDMA_CTRL)

Table 818. SDMA_CTRL field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 SDMA_ENn (n = 15 to 0)	Enable DMA for Slow Serial Messages on Channels 0 to 15. These bits are writeable when corresponding Slow Serial Message Ready Interrupt Enable bits are set to 0. 0 – DMA for Slow Serial Messages is disabled 1 – DMA for Slow Serial Messages is enabled

49.3.2.10 Fast Message Ready Interrupt Control Register (FRDY_IE)

Note: The following register figure and table show the maximum possible configuration but the exact number of valid register bits is dependent on the supported SENT channels on the device. See the Device Configuration chapter for the number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are unadvisable and may cause unexpected behavior.

Address Offset: 0x0034 Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	FRDY_IE15	FRDY_IE14	FRDY_IE13	FRDY_IE12	FRDY_IE11	FRDY_IE10	FRDY_IE9	FRDY_IE8
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	FRDY_IE7	FRDY_IE6	FRDY_IE5	FRDY_IE4	FRDY_IE3	FRDY_IE2	FRDY_IE1	FRDY_IE0
W								
Reset	0	0	0	0	0	0	0	0

Figure 799. Fast Message Ready Interrupt Enable Register (FRDY_IE)

Table 819. FRDY_IE field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 FRDY_IEn (n = 15 to 0)	Enable for Fast Message Ready Interrupt on Channels 0 to 15. These bits are writeable when corresponding DMA enable bits are set to 0 in the Fast Message DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Fast Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit is asserted in this register, an interrupt for corresponding channel is generated. 0 – Interrupt on reception of Fast Messages is disabled 1 – Interrupt on reception of Fast Messages is enabled

49.3.2.11 Slow Serial Message Ready Interrupt Enable Register (SRDY_IE)

Note: The following register figure and table show the maximum possible configuration but the exact number of valid register bits is dependent on the supported SENT channels on the device. See the Device Configuration chapter for the number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are unadvisable and may cause unexpected behavior.

Address Offset = 0x0038 Access: RW

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	SRDY_IE15	SRDY_IE14	SRDY_IE13	SRDY_IE12	SRDY_IE11	SRDY_IE10	SRDY_IE9	SRDY_IE8
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	SRDY_IE7	SRDY_IE6	SRDY_IE5	SRDY_IE4	SRDY_IE3	SRDY_IE2	SRDY_IE1	SRDY_IE0
W								
Reset	0	0	0	0	0	0	0	0

Figure 800. Slow Serial Message Ready Interrupt Enable Register (SRDY_IE)

Table 820. SRDY_IE field descriptions

Field	Description
0:15	Reserved. Read returns zero
16:31 SRDY_IEn (n = (CH-1) to 0)	Enable for Slow Serial Message Ready Interrupt on Channels 0 to 15. These bits are writeable when corresponding DMA enable bits are set to 0 in the Slow Serial DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Slow Serial Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit in this register is asserted, an interrupt for corresponding channel is generated. 0 – Interrupt on reception of Slow Serial Messages is disabled 1 – Interrupt on reception of Slow Serial Messages is enabled

49.3.2.12 DMA Fast Message Data Read Register (DMA_FMSG_DATA)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_DATA) via DMA.

Address Offset: 0x0040 Access: R

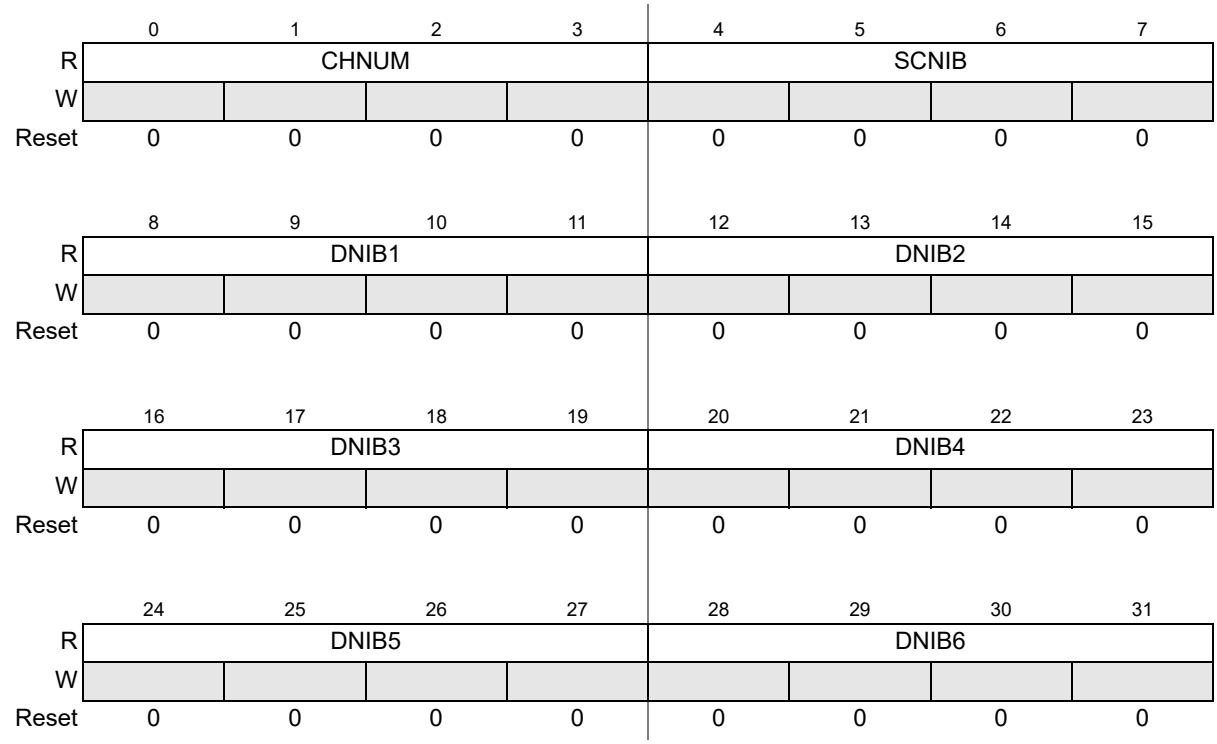


Figure 801. DMA Fast Message Data Read Register (DMA_FMSG_DATA)

Table 821. DMA_FMSG_DATA field descriptions

Field	Description
0:3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to ‘15’
4:7 SCNIB	Status and Communication Nibble of message
8:11 DNIB1	Data Nibble 1. Always supported
12:15 DNIB2	Data Nibble 2. Should be ignored if number of data nibbles supported by channel is 1
16:19 DNIB3	Data Nibble 3. Should be ignored if number of data nibbles supported by channel is 2 or less
20:23 DNIB4	Data Nibble 4. Should be ignored if number of data nibbles supported by channel is 3 or less
24:27 DNIB5	Data Nibble 5. Should be ignored if number of data nibbles supported by channel is 4 or less
28:31 DNIB6	Data Nibble 6. Should be ignored if number of data nibbles supported by channel is 5 or less

49.3.2.13 DMA Fast Message CRC Read Register (DMA_FMSG_CRC)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_CRC) via DMA.

Address Offset = 0x0044 Access: R

	0	1	2	3	4	5	6	7
R								
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R					CRC4b			
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R								
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R								
W								
Reset	0	0	0	0	0	0	0	0

Figure 802. DMA Fast Message CRC Read Register (DMA_FMSG_CRC)

Table 822. DMA_FMSG_CRC field descriptions

Field	Description
0:11	Reserved. Read returns zeros.
12:15 CRC4b	4-bit CRC value of the message.
16:31	Reserved. Read returns zeros.

49.3.2.14 DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the *Global Control Register (GBL_CTRL)* register. DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_TS) via DMA.

Address Offset: 0x0048 Access: R

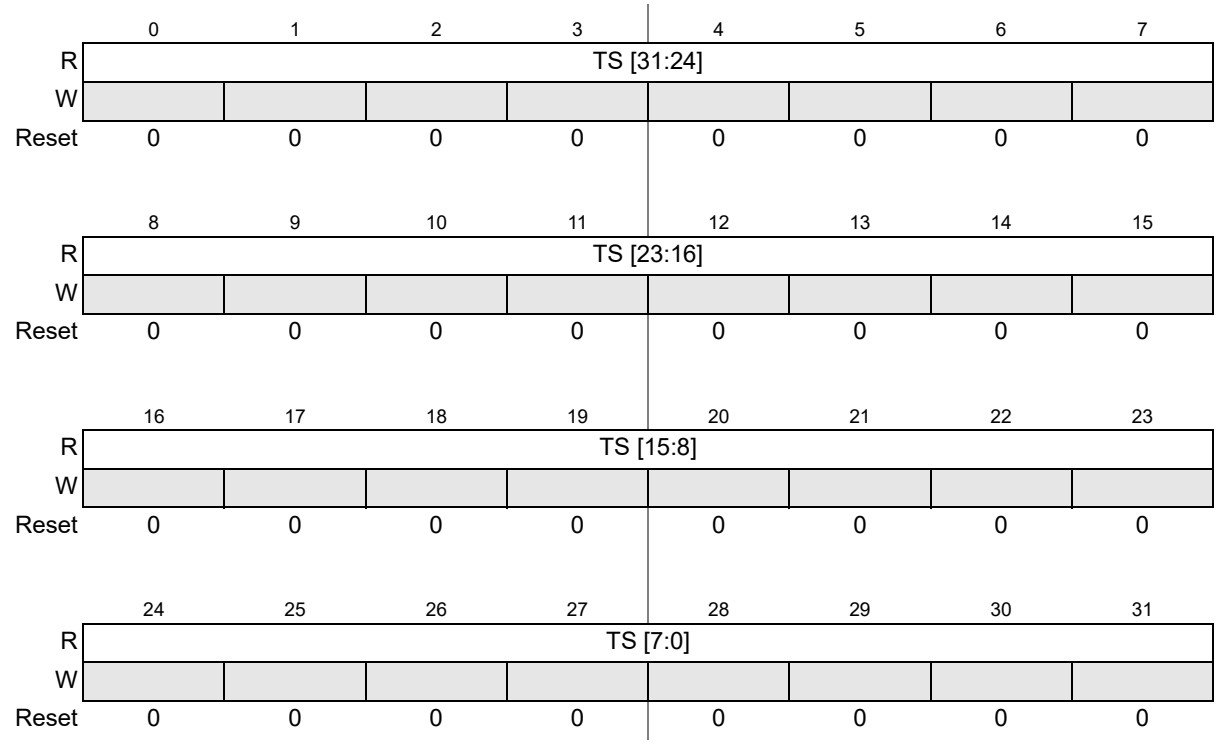


Figure 803. DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)

Table 823. DMA_FMSG_TS field descriptions

Field	Description
0:31 TS	Time Stamp for received message. Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

49.3.2.15 DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)

The above register (DMA_SMSG_BIT3) is common for all types of slow serial messages. Bit C defines the type of enhanced packet and bit TYPE defines whether it is short serial or enhanced serial message. Since this register is used to read both Short and Enhanced Serial Messages, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown in *Figure 812*, *Figure 813* and *Figure 814*. *Figure 812* and *Figure 813* show the enhanced serial messages format to be used when reading this register for these two types of message. *Figure 814* shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Address Offset = 0x0050 Access: R

	0	1	2	3	4	5	6	7
R	CHNUM				TYPE			
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R								
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R					0	CFG	ID[7:6] / ID[3:2]	
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	ID[5:4] / ID[1:0]		0	ID[3:0] / DATA[15:12]				0
W								
Reset	0	0	0	0	0	0	0	0

Figure 804. DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)

Table 824. DMA_SMSG_BIT3 Field Descriptions

Field	Description
0:3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to ‘15’
4 TYPE	Serial Message Type. Indicates the type of Serial Message received 0 – Short Serial Message 1 – Enhanced Serial Message
5:19	Reserved. Read returns zeros
20	Placed as 0 by Specification
21 CFG	Configuration bit ‘C’. Indicates the type of Enhanced Serial message 0 – Enhanced Serial Message with 8-bit ID field 1 – Enhanced Serial Message with 4-bit ID field
22:25 ID[7:4]/ID[3:0]	ID Field. Value depends on setting of ‘C’ bit. If ‘C’ bit is 0, this field is ID[7:4] and if the ‘C’ bit is 1, this field is ID[3:0]
26	0 as per SAE Specification

Table 824. DMA_SMSG_BIT3 Field Descriptions(Continued)

Field	Description
7:30 ID[3:0]/DATA [15:12]	ID or Data Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12]
31	0 as per SAE Specification

Note: If the TYPE bit in the register reads '0', then remainder bits (for CFG, ID & DATA) will be redundant and read zeros. The short serial message will be located in DMA_SMSG_BIT2 register fields. If TYPE bit is '1', then the message in register is Enhanced Serial Message and using CFG bit, software can decode the ID & DATA fields.

49.3.2.16 DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)

This register (DMA_SMSG_BIT2) is common for all types of slow serial messages. Hence, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown in *Figure 812*, *Figure 813* and *Figure 814*. *Figure 812* and *Figure 813* shows the enhanced serial messages format to be used when reading this register for these two types of messages. *Figure 814* shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Address Offset: 0x0054 Access: R

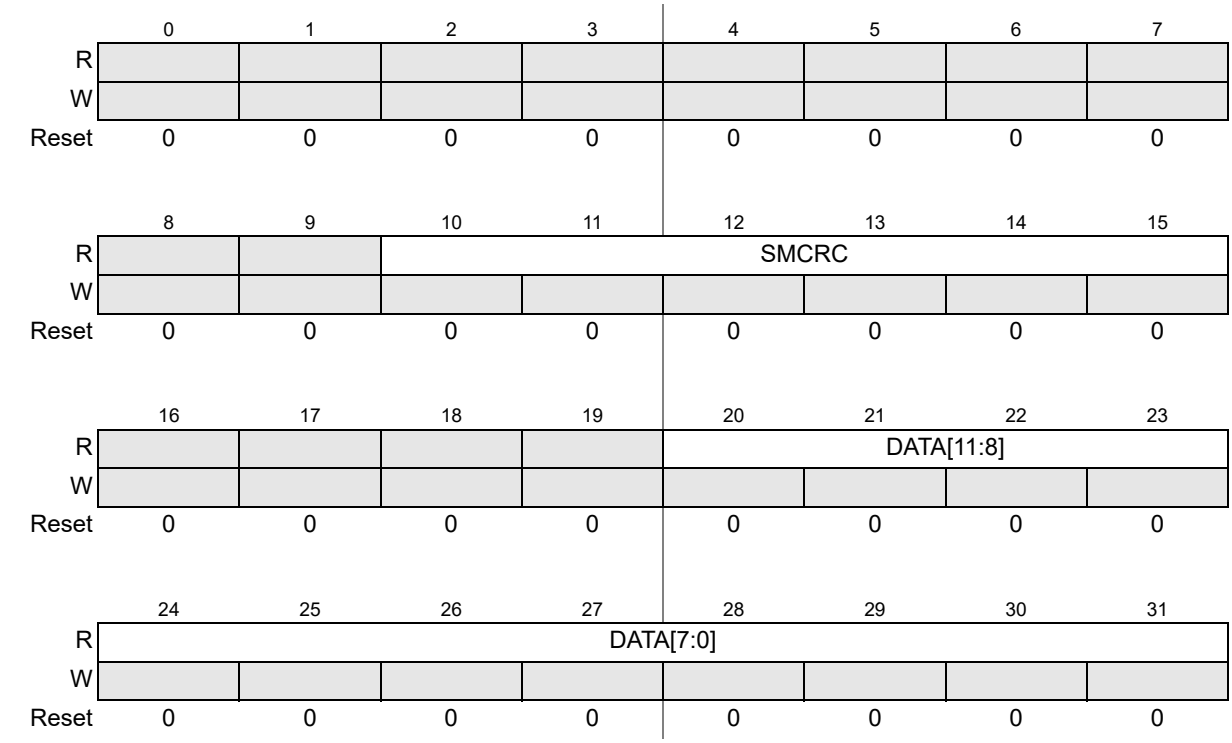


Figure 805. DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)

Table 825. DMA_SMSG_BIT2 field descriptions

Field	Description
0:9	Reserved. Read returns zeros.
10:15 SMCRC	6-bit CRC value of the message. When TYPE bit in *Section 49.3.2.15: DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)* is set to 0, this CRC is of 4 bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
16:19	Reserved. Read returns zeros.
20:31 DATA[11:0]	12-bit Data Value

49.3.2.17 DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the *Section 49.3.2.1: Global Control Register (GBL_CTRL)* register.

DMA Source Size must be set to 32 bits when reading this register (DMA_SMSG_TS) via DMA.

Address Offset: 0x0058 Access: R

	0	1	2	3	4	5	6	7
R	TS [31:24]							
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	TS [23:16]							
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	TS [15:8]							
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	TS [7:0]							
W								
Reset	0	0	0	0	0	0	0	0

Figure 806. DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)

Table 826. DMA_SMSG_TS field descriptions

Field	Description
0:31 TS	Time Stamp for the received message. Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

49.3.2.18 Channel ‘n’ Clock Control Register (n = 0 to (CH-1)) (CHn_CLK_CTRL)

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

The SENT module generates the Receiver Sensor Clock (or simply Receiver Clock) internally to sample the incoming filtered sensor data. The user software must program the prescaler value (for Receiver Clock) to match the ideal Sensor Tx Clock period. The generated Receiver Clock is compensated for variations, clock drift and jitter that might occur in the Sensor Tx Clock. See *Section 49.4.5: Adjustment for variation in sensor (Tx) clock on page 1368* for more details.

This register configures the generation and subsequent compensation logic of the Receiver Clock. User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the *Section 49.3.2.2: Channel Enable Register (CHNL_EN)*.

Address Offset = 0x0060 + n × 0x10 Access: RW

	0	1	2	3	4	5	6	7
R		CM_PRSC[14:8]						
W								
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	CM_PRSC[7:0]							
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	COMP_EN		PRSC[13:8]					
W								
Reset	1	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	PRSC[7:0]							
W								
Reset	0	0	0	0	0	0	0	0

Figure 807. Channel ‘n’ Clock Control Register (CHn_CLK_CTRL)

Table 827. CHn_CLK_CTRL field descriptions

Field	Description
0	Reserved. Read returns zero
1:15 CM_PRSC	Compensated Prescaler Value. This is the compensated prescaler value calculated by Clock Compensation logic and this value is used to compensate for sensor Tx clock variation. This value is indicated as a read only field for use by user software, if needed. This is a 15-bit value. '= Rx Prescaler Value' – No variation, hence no compensation or Compensation is disabled '< Rx Prescaler Value' – Tx Clock is SLOWER than the programmed receive clock '> Rx Prescaler Value' – Tx Clock is FASTER than the programmed receive clock Thus, the new Receiver Clock Tick Period can be computed by software as *Compensated Prescaler Value* $\times$ *High Frequency Receiver Clock Period*
16 COMP_EN	Compensation Enable. To enable compensation logic to adjust the receiver clock against variation in Tx clock on this channel. User must set the Prescaler Value before enabling this bit. 0 – Compensation is disabled 1 – Compensation is enabled
17	Reserved. Read returns zero
18:31 PRSC	Rx Prescaler Value. This factor is required to generate required Receiver Clock tick from the High Frequency Receiver Clock. This value can only be written when Channel Enable bit is Zero. Hence this value must be programmed before enabling the channel. If required receiver clock tick period (T_{rx_clk}) is not divisible by the high frequency receiver clock (protocol clock) period (T_{hf_clk}), the required prescaler value will have a fractional part which cannot be programmed. In this case then user is advised to program the prescaler value to the lower integer value. 0 – Prescaler is bypassed. Should not be used. Non-Zero Value – Enables Prescaler The required value is computed by software as follows: **Equation 33** $$\text{Prescaler_value} = \text{floor}\left(\frac{T_{rx_clk}}{T_{hf_clk}}\right)$$ (both clock periods taken in µs). See *Section 49.5.3: High frequency receiver clock (protocol clock) requirements on page 1374* before setting this value. For example, if the required receiver clock tick period is 43 µs and the High Frequency Receiver clock is selected to be 62.5 MHz (or 0.016 µs) then, Prescaler Value = 43/0.016 = 2687.5. Hence user must program 2687 in this register.

49.3.2.19 Channel 'n' Status Register (n = 0 to (CH-1)) (CHn_STATUS)

This registers stores information about certain events and error conditions that occur in the channel during the message reception process. The assertion of these bit is not linked to any message but it is a general alert to the CPU that a particular condition (error or normal

event) happened in the recent past. Messages are automatically discarded when an error is detected.

User software can clear these bit at any time by writing 1 to them. This register can be updated at any time irrespective of the setting in the *Section 49.3.2.2: Channel Enable Register (CHNL_EN)* bit for the corresponding channel.

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

Address Offset = 0x0064 + n × 0x10 Access: W1C

	0	1	2	3	4	5	6	7
R	BUS_IDLE				CAL_RESYNC	CAL_20_25	SMSG_OFLW	FMSG_OFLW
W	w1c				w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R		PP_DIAG_ERR	CAL_LEN_ERR	CAL_DIAG_ERR	NIB_VAL_ERR	SMSG_CRC_ERR	FMSG_CRC_ERR	NUM_EDGES_ERR
W		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R								
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R								
W								
Reset	0	0	0	0	0	0	0	0

Figure 808. Channel ‘n’ Status Register (CHn_STATUS)

Table 828. CHn_STATUS field descriptions

Field	Description
0 BUS_IDLE	Bus Idle Status. This bit indicates that the sensor interface has been idle for more than the period defined by the programmed Bus Idle Count value (in *Section 49.3.2.20: Channel ‘n’ Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)*). CPU should write 1 to this clear this bit once it is read. 0 – Bus is not idle 1 – Channel has been idle for more than the allowed value
1:3	Reserved. Read returns zero

Table 828. CHn_STATUS field descriptions(Continued)

Field	Description
4 CAL_RESYNC	Successive Calibration Check (Option 2 of SAE SENT Spec) Resynchronized. This bit indicates that Successive Calibration Check (Option 2 of SAE Spec) has failed three times and has resynchronized to assume 3rd message to be correct and will be received. The successive calibration check error will also be asserted during re-synchronization (to indicate the 3rd error being detected). 0 – No interrupt 1 – Interrupt Status condition has occurred
5 CAL_20_25	Calibration Variation 20 – 25% Interrupt Status. This bit indicates that the calibration pulse received on this channel has variation in between ±20% to ±25% from 56 ticks. 0 – No Interrupt 1 – Interrupt Status condition has occurred
6 SMSG_OFLW	Slow Serial Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow will cause the older messages (which are not read out) to be overwritten with newly received messages. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 – No Interrupt 1 – Interrupt Status condition has occurred
7 FMSG_OFLW	Fast Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow happens when effective data read rate by CPU is slower than data receive rate on channel. On overflow, new incoming messages will overwrite the previous messages (that are not read out) stored in the receiver buffers. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 – No Interrupt 1 – Interrupt Status condition has occurred
8	Reserved. Read returns zero

Table 828. CHn_STATUS field descriptions(Continued)

Field	Description
9:15	Diagnostic Checks Status bits. These bits indicate which diagnostic has failed or passed on this channel. These bits are cleared by writing 1 to them. NUM_EDGES_ERR: Not the expected number of negative edges between calibration pulse. This bit will not set in case of “Option 2” for successive calibration pulse check is selected. This error will only set when “Option 1” for successive calibration check is selected (SUCC_CAL_CHK). FMSG_CRC_ERR: Checksum error in Fast Message SMSG_CRC_ERR: Checksum error in Slow Serial Message NIB_VAL_ERR: Any nibble data value <0 or >15 CAL_DIAG_ERR: Successive Calibration pulses differ by ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the frequency of protocol clock (See *Section 49.5.3: High frequency receiver clock (protocol clock) requirements on page 1374*). CAL_LEN_ERR: This bit indicates that Calibration pulse is more than 56 ticks ±25%. PP_DIAG_ERR: This bit indicates that the ratio of calibration pulse length to overall message length (with pause pulse) is more than ±1.5625% between two messages. This check is valid only for messages with pause pulse. 0 – Error Check has passed 1 – Error Check has failed
15:016:31	Reserved. Read returns zero

49.3.2.20 Channel ‘n’ Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)

This register (CHn_CONFIG) provides the configurability for interrupt assertion for various events and error as explained in *Section 49.3.2.19: Channel ‘n’ Status Register (n = 0 to (CH-1)) (CHn_STATUS)* and control bits for selection of various receive options that might be required by the user software. This register also configures the width of noise glitches (on the sensor input) to be filtered out by the input programmable filter.

User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the *Section 49.3.2.2: Channel Enable Register (CHNL_EN)*.

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

Address Offset = 0x0068 + n × 0x10 Access: RW

	0	1	2	3	4	5	6	7
R W	BUS_IDLE_CNT				IE_CAL_RESYNC	IE_CAL_20_25	IE_SMSG_OFLW	IE_FMSG_OFLW
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R W	FCRC_CHK_OFF	IE_PP_DIAG_ERR	IE_CAL_LEN_ERR	IE_CAL_DIAG_ERR	IE_NIB_VAL_ERR	IE_SMSG_CRC_ERR	IE_FMSG_CRC_ERR	IE_NUM_EDGES_ERR
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R W	DCHNG_INT	CAL_RNG	PP_CHKSEL	FCRC_TYPE	FCRC_SC_EN	SCRC_TYPE	PAUSE_EN	SUCC_CAL_CHK
Reset	0	0	0	0	0	0	0	1

	24	25	26	27	28	29	30	31
R W	FIL_CNT[7:0]							
Reset	0	0	0	0	0	1	0	0

Figure 809. Channel ‘n’ Configuration Register (CHn_CONFIG)

Table 829. CHn_CONFIG field descriptions

Field	Description
0:3 BUS_IDEL_CNT	Bus Idle Count: This register value defines the maximum allowable idle period on the sensor interface of this channel. The value is defined as follows: 0000: Disabled. 0001: 127*2 Receiver Clock Tick Counts 0010: 127*4 Receiver Clock Tick Counts 0100: 127*8 Receiver Clock Tick Counts 1000: 127*16 Receiver Clock Tick Counts
4 IE_CAL_RESYNC	Successive Calibration Check Resynchronized Interrupt Enable: This bit enables interrupt assertion when Successive Calibration diagnosis has failed three times in case of “Option 2” being selected for Successive Calibration Check Method and the check resynchronizes the take the third message to be correct. 0 – Interrupt is disabled 1 – Interrupt is enabled
5 IE_CAL_20_25	Calibration Variation 20 - 25% Interrupt Enable. This bit enables interrupt assertion when corresponding status bit is set. 0 – Interrupt is disabled 1 – Interrupt is enabled

Table 829. CHn_CONFIG field descriptions(Continued)

Field	Description
6 IE_SMSG_OFLW	Slow Serial Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Slow Serial Messages in corresponding channel 0 – Interrupt is disabled 1 – Interrupt is enabled
7 IE_FMSG_OFLW	Fast Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Fast Messages in corresponding channel. 0 – Interrupt is disabled 1 – Interrupt is enabled
8 FCRC_CHK_OFF	Fast Message CRC Check Off: This bit can be used to switch off CRC check in Fast Message 0 – Check is enabled 1 – Check is disabled/off
9:15	Diagnostic Checks Interrupt Enable bits. These bits enables interrupts assertion when respective diagnostic checks fail on corresponding channel. IE_NUM_EDGES_ERR: Not the expected number of negative edges between calibration pulse IE_FMSG_CRC_ERR: Checksum error in Fast Message IE_SMSG_CRC_ERR: Checksum error in Slow Serial Message IE_NIB_VAL_ERR: Any nibble data value <0 or >15 IE_CAL_DIAG_ERR: Successive Calibration pulses differ by more than ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the protocol clock. For instance for 62.5 MHz, this check will fail for 1.56% and passes for 1.55% IE_CAL_LEN_ERR: Calibration pulse is wider than 56 ticks ±25% IE_PP_DIAG_ERR: Ratio of calibration pulse length to message length varies by more than ±1.5625% between two frames. Valid for messages with pause pulse. Configuration option for each bit as below: 0 – Interrupt is disabled 1 – Interrupt is enabled
16 DCHNG_INT	Enable for Interrupt on Reception of Fast Message with Changed Data: This bit is used to enable storing and generating corresponding DMA or Message Ready Interrupt Request for Fast Message only when any of the data nibbles after different from previously received Fast Message on that channel. If a new Fast Message is received that has same values for all data nibbles as that of previous message then the current message will be discarded and will not be stored in the buffer, and neither Fast Message Ready Interrupt nor DMA request will be generated. 0 – All Fast Messages will be received even if data nibbles are same 1 – Only Fast Messages with differing values of data nibbles will be received
17 CAL_RNG	Valid Calibration Pulse Range Selection: This bit is used to control whether 20% variation or 25% variation in Calibration Pulse will be tolerated when detecting a Calibration Pulse 0 – 20% variation is acceptable 1 – 25% variation is acceptable

Table 829. CHn_CONFIG field descriptions(Continued)

Field	Description
18 PP_CHKSEL	Pause Pulse Diagnostic Check Selection: This bit controls which diagnostic checks to run when messages are enabled to be received with pause pulses. 0 – Both Successive Calibration Pulse Check and Pause Pulse Diagnostic are run 1 – Only Pause Pulse Diagnostic is run. Successive Calibration Pulse Check is disabled.
19 FCRC_TYPE	Fast Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received fast messages 0 – XOR based implementation (Recommended by SAE Specification) 1 – Legacy LUT based implementation
20 FCRC_SC_EN	Fast Message CRC Status and Communication Nibble Enable. This bit enables the inclusion of Status and Communication Nibble when CRC is calculated for Fast Messages. 0 – Status and Communication Nibble not included in CRC calculation 1 – Status and Communication Nibble is included in CRC calculation
21 SCRC_TYPE	Slow Serial Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received slow serial messages. 0 – XOR based implementation (Recommended by SAE Specification) 1 – Legacy LUT based implementation
22 PAUSE_EN	Pause Pulse Enable. Enables the channel receiver to detect a pause pulse. 0 – Detection of Pause Pulse is disabled 1 – Detection of Pause Pulse enabled
23 SUCC_CAL_CHK	Successive Calibration Pulse Check Method. This bit indicates the method to be used for performing the successive calibration pulse check. Default value is 1. 0 – Option 2 i.e. Low Latency Option as per SAE Specification 1 – Option 1 i.e. Preferred but High Latency Option as per SAE Specification
24:31 FIL_CNT	Input Filter Sample Count. This indicates the number of Protocol Clock (of High Frequency Receiver Clock) cycles required for channel input from device's pad to remain stable before it is sampled as 0 or 1. This field defines the width of glitches to be filtered out by the input programmable filter on that channel. Only one bit should be set in this field and the output of filter will be offset by twice the same number of clocks plus additional 3 clocks due to synchronization. Default value is '4'. 0 – No filtering. Input from device's pad is only synchronized Non-Zero – Filtering is enabled

49.3.2.21 Channel 'n' Fast Message Data Read Register (n = 0 to (CH-1)) (CHn_FMSG_DATA)

Address offset: 0x0160 + 0x18 × n

This is register CHn_FMSG_DATA. The contents of this register are same as *Section 49.3.2.12: DMA Fast Message Data Read Register (DMA_FMSG_DATA)*.

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

49.3.2.22 Channel ‘n’ Fast Message CRC Read Register (n = 0 to (CH-1)) (CHn_FMSG_CRC)

Address offset: 0x0164 + 0x18 × n

This is register CHn_FMSG_CRC. The contents of this register are same as *Section 49.3.2.13: DMA Fast Message CRC Read Register (DMA_FMSG_CRC)*.

49.3.2.23 Channel ‘n’ Fast Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_FMSG_TS)

Address offset: 0x0168 + 0x18 × n

This is register CHn_FMSG_TS. The contents of this register are same as *Section 49.3.2.14: DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)*.

49.3.2.24 Channel ‘n’ Serial Message Read Register (Bit 3) (n = 0 to (CH-1)) (CHn_SMSG_BIT3)

Address offset: 0x016C + 0x18 × n

This is register CHn_SMSG_BIT3. The contents of this register are same as *Section 49.3.2.15: DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)*.

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

49.3.2.25 Channel ‘n’ Serial Message Read Register (Bit 2) (n = 0 to (CH-1)) (CHn_SMSG_BIT2)

Address offset: 0x0170 + 0x18 × n

This is register CHn_SMSG_BIT2. The contents of this register are same as *Section 49.3.2.16: DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)*.

49.3.2.26 Channel ‘n’ Serial Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_SMSG_TS)

Address offset: 0x0174 + 0x18 × n

This is register CHn_SMSG_TS. The contents of this register are same as *Section 49.3.2.17: DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)*.

49.3.3 Register Classification for Safety

The registers of this module are classified as per Safety Requirements as follows:

Safety Critical: There are no safety critical registers in this module.

Safety Relevant: The register bits/fields in this category are safety relevant as any random error in these bits will lead to unwanted interference to CPU/DMA. The registers containing

these bits must be protected by periodic CRC or equivalent check implemented at SoC Level. The following register bits/fields are classified in this category:

- *Section 49.3.2.1: Global Control Register (GBL_CTRL)*
- *Section 49.3.2.2: Channel Enable Register (CHNL_EN)*
- *Section 49.3.2.4: Fast Message Ready Status Register (FMSG_RDY)*
- *Section 49.3.2.5: Slow Serial Message Ready Status Register (SMSG_RDY)*
- *Section 49.3.2.6: Data Control Register 1 (DATA_CTRL1)* and *Section 49.3.2.7: Data Control Register 2 (DATA_CTRL2)*
- *Section 49.3.2.8: Fast Message DMA Control Register (FDMA_CTRL)*
- *Section 49.3.2.9: Slow Serial Message DMA Control Register (SDMA_CTRL)*
- *Section 49.3.2.10: Fast Message Ready Interrupt Control Register (FRDY_IE)*
- *Section 49.3.2.11: Slow Serial Message Ready Interrupt Enable Register (SRDY_IE)*
- Fast Message DMA Read Register Set and Slow Message DMA Read Register Set

Safety Latent: The register bits/fields in this category should be protected by LBIST at SoC level. The following register bits/fields are classified in this category:

- *Section 49.3.2.3: Global Status Register (GBL_STATUS)*
- *Section 49.3.2.19: Channel ‘n’ Status Register (n = 0 to (CH-1)) (CHn_STATUS)*
- *Section 49.3.2.20: Channel ‘n’ Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)*

Non-Safety: The following register bits are classified in this category:

- All other registers and bits fall in this category

49.4 Functional description

49.4.1 Initialization sequence

All the channels are required to be configured before enabling them by writing into corresponding channel enable bit in the *Section 49.3.2.2: Channel Enable Register (CHNL_EN)*. There are some registers which must be configured and some which are to be changed if reset value is not as required for that channel. After enabling the channel, no configuration should be changed and if required to be changed, the user needs to first disable the channel and then reconfigure the channel parameters. The following must to be configured before enabling the corresponding channel:

- Time Stamp Prescaler Value in *Section 49.3.2.1: Global Control Register (GBL_CTRL)* to generate the time stamping clock
- Rx Prescaler Value in *Section 49.3.2.18: Channel ‘n’ Clock Control Register (n = 0 to (CH-1)) (CHn_CLK_CTRL)* to generate the channel’s receiver clock
- Compensation enable in *Section 49.3.2.18: Channel ‘n’ Clock Control Register (n = 0 to (CH-1)) (CHn_CLK_CTRL)*
- Channel Configuration settings in the *Section 49.3.2.20: Channel ‘n’ Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)*
- Number of Data Nibbles in Channel in the *Section 49.3.2.6: Data Control Register 1 (DATA_CTRL1)* and *Section 49.3.2.7: Data Control Register 2 (DATA_CTRL2)*
- Control bits that control whether messages received in channel are to read by interrupt or DMA in *Section 49.3.2.8: Fast Message DMA Control Register (FDMA_CTRL)*, *Section 49.3.2.9: Slow Serial Message DMA Control Register (SDMA_CTRL)*,

Section 49.3.2.10: Fast Message Ready Interrupt Control Register (FRDY_IE) and *Section 49.3.2.9: Slow Serial Message DMA Control Register (SDMA_CTRL)*

49.4.2 DMA read logic

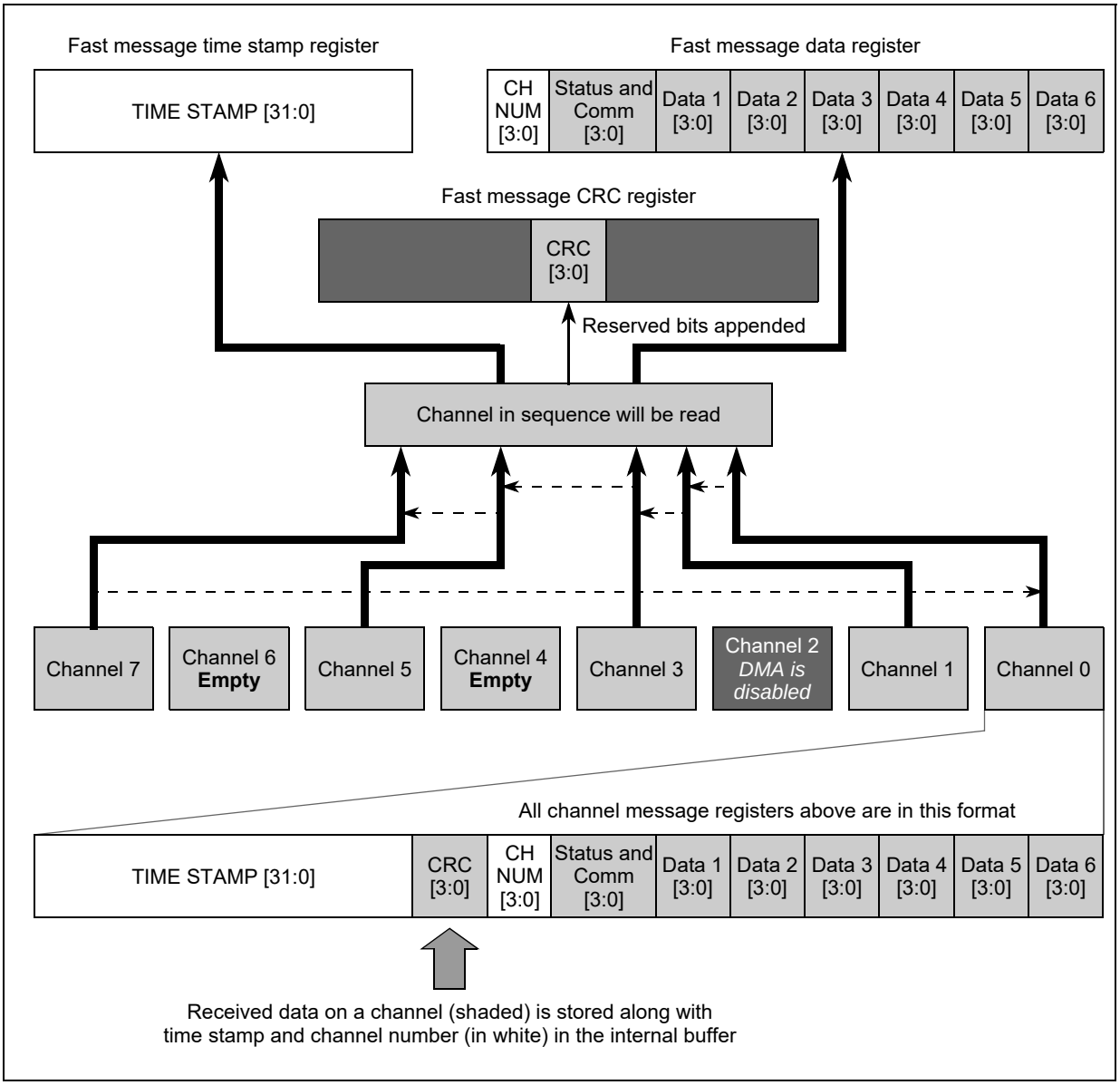
The SENT module generates separate DMA requests for all Fast Messages and Slow Serial Messages received. The DMA requests are common for all channels. Each packet stored in the channels' buffers are appended with a time stamp taken from a free running time keeping counter and the channel number on which it was received. Details on this time stamp counter can be found in *Section 49.4.8: Time stamp logic on page 1371*.

49.4.2.1 DMA request for Fast Message reading

Fast Messages received are stored in a single system bus clock domain buffer to be read out by the DMA Controller via the DMA Fast Message Read register set. Note the DMA Controller must read all 3 DMA Read Registers to obtain the complete message. The data flow will happen as follows:

- A DMA request will be asserted when any of the DMA enabled channels has successfully received one complete Fast Message. The request is kept asserted till all messages (across DMA enabled channels) have been read out.
- DMA Controller should read the complete message by accessing each register of the DMA Fast Message Read register set, the registers being:
  - *Section 49.3.2.12: DMA Fast Message Data Read Register (DMA_FMSG_DATA)*
  - *Section 49.3.2.13: DMA Fast Message CRC Read Register (DMA_FMSG_CRC)*
  - *Section 49.3.2.14: DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)*
- A fixed round robin sequencing runs, as shown in *Figure 810*, and this logic checks for messages in sequence 0→1→3 →5→7 and back to channel 0 and so on. Channel 2 is skipped as it is not DMA enabled) and channel 4 and 6 are skipped as message is not received completely. This sequence continues like this.
- The sequence in which messages are read through DMA interface is not deterministic and depends on the sequence in which they are completely received.
- When the sequencing logic detects a message ready to be read, it will update the DMA Fast Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 read accesses to the above mentioned registers)
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- User software should ensure there is no overflow in channels. In case overflow occurs, the message in the channel buffer will get overwritten by the newly received message. However, when a DMA read is in progress, the buffers will not be overwritten if overflow occurs to maintain data integrity.
- The Round Robin sequencing logic loops through each DMA enabled channel and halts at any channel having data till it is read out by the DMA Controller.

Figure 810. Fast Message DMA read logic



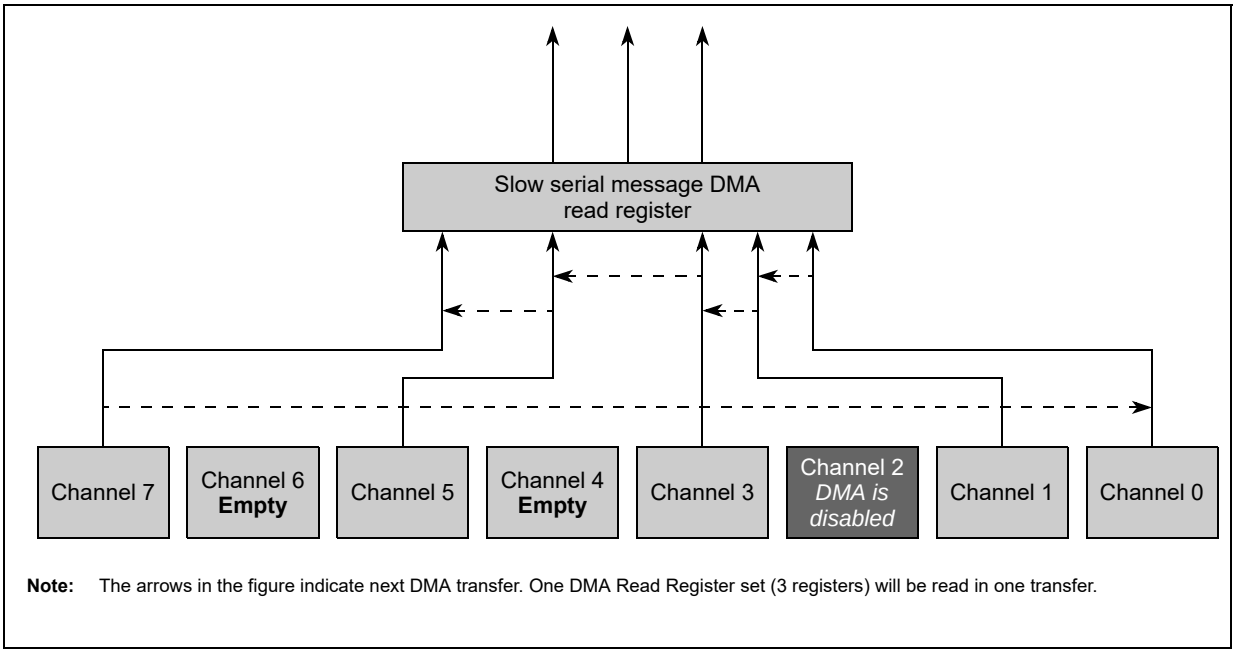
49.4.2.2 DMA request for Slow Serial Message reading

A separate DMA request (different from DMA request for Fast Message) is generated for all slow serial messages received on all channels. The data flow will happen as follows:

- Request will be asserted when any of the DMA enabled channels has successfully received one complete serial message (short or enhanced) and kept asserted till all messages have been read from their respective memory-mapped registers
- Similar to DMA read for Fast Messages, data for slow serial messages will be read from DMA Slow Serial Message Read register set in memory map, the register set comprising of:
  - *Section 49.3.2.15: DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)*
  - *Section 49.3.2.16: DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)*
  - *Section 49.3.2.17: DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)*
- A fixed round robin sequencing logic will be running and it will update the DMA Slow Serial Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 DMA read accesses to the above mentioned registers)
- The channel number and time stamp will also be appended for every message
- A message type bit (TYPE) will also be appended to distinguish between short and enhanced serial messages
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- Since serial messages receive rate is very slow, user software should ensure there is no overflow
- The fixed round robin sequence logic for DMA read is the same as that used for Fast Messages (see *Section 49.4.2.1: DMA request for Fast Message reading*)

Figure 811 shows the sequencing of channels for serial messages to be read when DMA read is done on DMA Slow Serial Message Read register set.

Figure 811. Slow Serial Message DMA read logic



Note: The arrows in the figure indicate next DMA transfer. One DMA Read Register set (3 registers) will be read in one transfer.

The formats in which the 3 types of Serial Messages will be stored in the buffers are shown in the figures below.

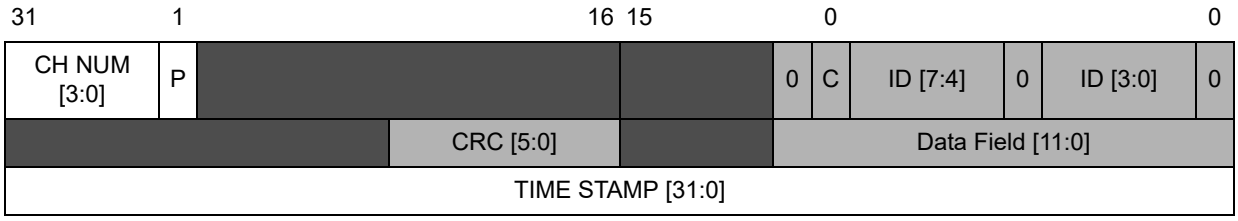


Figure 812. Enhanced Serial Message with 8-bit ID field

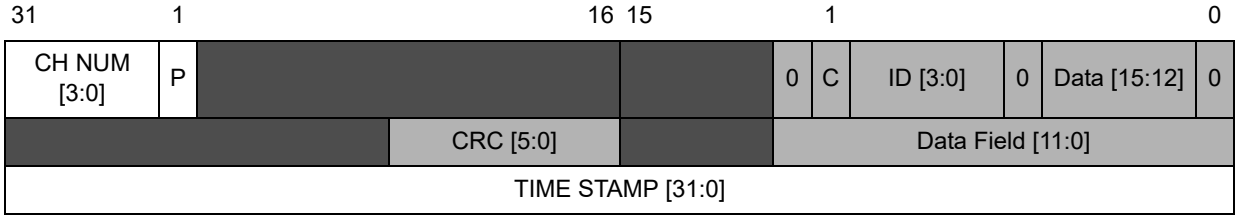


Figure 813. Enhanced Serial Message with 4-bit ID field

Figure 812 and *Figure 813* shows the enhanced serial messages. Bit C (shown in figure) or bit CFG (as shown in *Section 49.3.2.15: DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)* description) defines the type of enhanced packet and bit P (shown in figure) or bit TYPE (shown in *Section 49.3.2.15: DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)* description) defines whether it is short serial or enhanced serial message. CRC, ID and Data fields are stored in format as received and shown in SAE specification with just one exception that CRC is shifted to align with 16-bit boundary.

Figure 814 shows the short serial message. Field locations have been placed in such a way to allow a common register to read all the slow serial messages and also match the bit positions in actual message defined by SAE Specification.

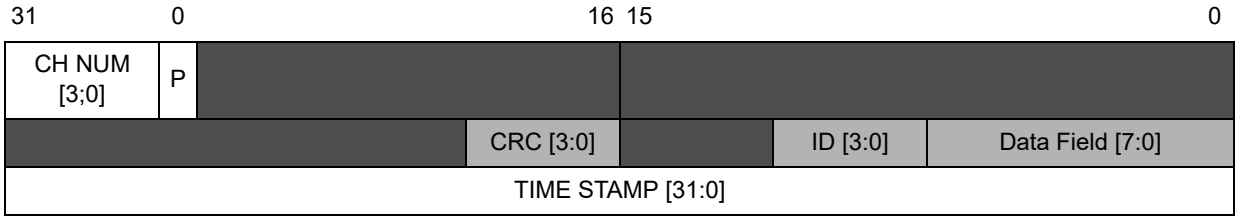


Figure 814. Short serial message

Again, three 32-bit accesses to be made by DMA to get one complete message with time stamp.

49.4.3 Message reading via interrupts

There is one interrupt request per channel for a Fast message received and one for a serial message received. Each channel is individually configurable to enable or disable interrupt on reception of Fast or Slow Serial messages separately. In case of interrupt request of Fast Message, an interrupt will be asserted when any of the channels has received a complete Fast Message without errors and that channel is not configured for DMA Reads. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing all channels that are enabled to give out messages via interrupts through the Register Interface.

In response to these interrupts, the CPU is expected to read interrupt status register for fast or Slow Messages (depending on which interrupt was asserted) and store it in system memory. Using this status register value, the interrupt routine can decide from which channels the message is to be read. For instance if bit corresponding to channel 0 is set then routine should read from the address location corresponding to Channel 0 Fast Message Read registers and move onto the next asserted bit. The interrupt routine should scan through all the bits of this stored interrupt status register value and read corresponding channel's fast or Slow Message registers. In this manner all available messages can be read. After scanning through all the bits of status register stored in system memory, routine can read again the message ready interrupt status register to find out if any other channel has received in the mean time. This will save another interrupt routine time mainly from stacking/unstacking delay.

It is assumed that the interrupt controller will be able to respond fast enough to avoid any overflow as only one message per channel will be stored and the system bus clock frequency is sufficient to avoid this.

49.4.3.1 Suggested software interrupt service routine steps

These are suggested step to have an interrupt routine that can read all messages quickly and avoid any underflow/overflow. Actual implementations can vary.

When separate Fast Message and Slow Serial Message Interrupts are used:

The Fast Message Interrupt should have higher priority than Slow Serial Message Interrupt. The suggested ISR steps for a Fast Message Interrupts can be

- Read the Fast Message Ready Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register set (3 registers). Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Fast Message Status Registers again to ensure no new message has arrived during the ISR.
- If there are no bits set, exit the ISR, else repeat this process.

The ISR steps for Slow Serial Message Interrupt can be the same as for Fast Message ISR.

When common Fast Message and Slow Serial Message Interrupts are used:

The suggested ISR steps can be

- Read the Fast Message Status Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Slow Serial Message Status Register into a new variable
- Shift out bits starting from MSB from this new variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Slow Serial Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, software can again check both Fast and Slow Serial Message Ready register to ready any message that might have been received while the routine was being serviced, or can choose to exit the routine.

49.4.4 Overflow behavior

This section describes the overflow behavior.

49.4.4.1 Fast messages buffers

Overflow in Fast Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, the newly received message overwrites the previously stored message and the overflow status bit is set. However, if overflow asserts while CPU/DMA is reading that channel (i.e. all 3 read registers have not yet been read), then that message is not overwritten. While messages are being overwritten, the slow message reception is not hampered.

49.4.4.2 Slow message buffers

Overflow in Slow Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, further reception of slow messages is halted until the buffers are read out by CPU or DMA. Since it is less likely that CPU or DMA do not read the read registers for 32 or 36 fast message length duration, an overflow in slow messages would not be as frequent as in fast messages.

49.4.5 Adjustment for variation in sensor (Tx) clock

The clock used by the Channel Receiver (referred to as Receiver Clock, or Rx_CLK) to sample the message nibbles is generated from a High Frequency Receiver Clock or Protocol Clock. This high frequency receiver clock is divided using a prescaler counter which generates a tick when it rollovers on reaching a count equal to the Prescaler Value programmed in the channel's clock control register. The receive clock is generated separately for each channel. By varying this Prescaler Factor, we can control the period of the receive clock and make it close to the Transmitter clock used by the sensor for the particular channel.

After reset and once channel is enabled, the compensated prescaler value is set to Zero and the user software programs the Prescaler Value to obtain the desired receiver clock frequency for the particular channel. Thus, the receiver starts working by assuming that the transmitter is sending messages on the frequency that is programmed in its channel clock control register. The channel's input after filtering is continuously measured by the Prescaler Counter. The compensation logic checks each pulse length and determines the calibration pulse from them. When a calibration pulse is detected, the compensated prescaler value is computed and stored in the Channel's Clock Control Register (*Section 49.3.2.18: Channel 'n' Clock Control Register (n = 0 to (CH-1)) (CHn_CLK_CTRL)*). Software can use this value to determine the variation in Tx Clock.

Note: A pause pulse may incidentally be detected as a synchronization/calibration pulse during start up after reset. The correction value would be computed on the pause pulse too but since the next synchronization pulse would follow, the logic would again detect the falling edge on the synchronization pulse and the correction value would now be correctly computed on the correct synchronization pulse.

The Clock Compensation or Correction happens for every messages that is received and the compensated prescaler value remains constant for that message.

49.4.5.1 Adjustment for nibble length variation

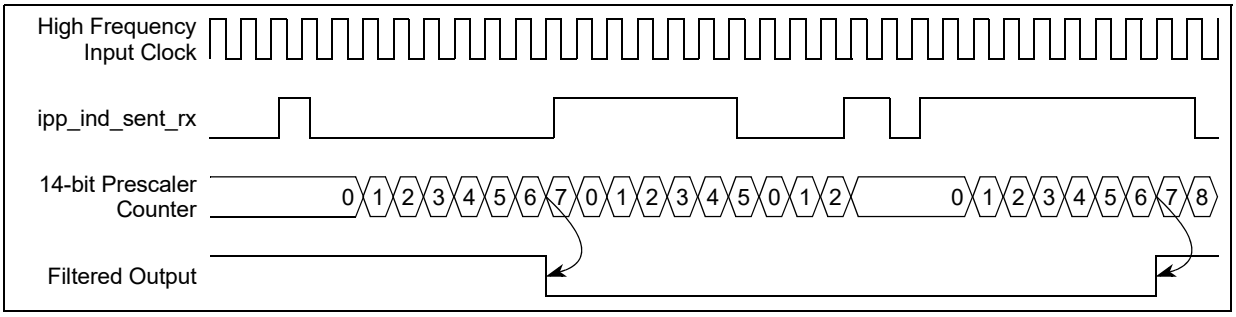
Section 6.2 of SAE Specifications specifies the maximum allowable limits for clock drift and jitter for the transmitter and receiver separately. This can cause the length of nibbles to vary and can cause incorrect sampling of data nibbles. This variation is adjusted by the SENT Receiver when nibbles are being sampled.

49.4.6 Input programmable filter

The Input Programmable Filter ensures that only valid input pin transitions are received by the SENT Receiver module. The input programmable filter is an 8-bit programmable up counter that increments on the high frequency receiver clock. The sensor input signal from device's pad is synchronized by high frequency receiver clock. When a state change occurs

in the sensor input signal, the 8-bit counter resets and starts counting up on the high frequency receiver clock. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflows occurs, the new pin value (i.e. sensor input) is validated and latched in the filtered output flop (also running on high frequency receiver clock). If the opposite edge appears on the sensor input before validation (i.e. overflow), the counter is reset and the input is not passed onto the filtered output. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the filter output. The number of samples (or the match value of the up counter) is programmable by the user in the channel's configuration register (*Section 49.3.2.20: Channel 'n' Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)*). When zero is programmed into the registers, it bypasses the filter and the synchronized input is output as the filtered output. A non-zero value delays the filtered output by the number of clocks proportional to the programmed value in the Channel Configuration Register (*Section 49.3.2.20: Channel 'n' Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)*). A timing diagram of the input filter is shown in *Figure 815*.

Figure 815. Input programmable filter timing diagram



49.4.7 Receiver diagnostics

Most of the receiver diagnostics are controlled by the Fast Message State Machine. The following checks are conducted by the receiver in each channel

- Calibration pulse length < 56 clock ticks – 25% or > 56 clock ticks + 25%
- Not the expected number of falling edges between calibration pulses. (Message length is pre-defined by each sensor device and programmed by user for each channel)
- Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use i.e. Legacy or XOR-based.
- Any nibble data values measured as < 0 or > 15
- Successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used.
- For messages with pause pulse, an additional diagnostic is done; however, this diagnostic will not fail any message (to reduce latency in message reception) but the failure of this diagnostic will be indicated in the channel's status register.

On detection of any of the above errors, the receiver rejects the current packet until a new calibration pulse is detected. Occurrence of an error is flagged in the respective channel's status register (*Section 49.3.2.19: Channel 'n' Status Register (n = 0 to (CH-1)) (CHn_STATUS)*). If error interrupt is enabled, an error interrupt will be generated. There is a single error interrupt generated for all errors, so the CPU must read the status register to find out which error occurred. Once error condition is flagged, subsequent errors (except NUM_EDGES_ERR) are automatically masked out until a valid calibration pulse is

detected, after which all checks are automatically enabled. This is done to avoid multiple assertions of the interrupt to CPU in case the channel become unstable and send wrong data continuously.

49.4.7.1 Calibration pulse length check

The calibration pulse length check happens in conjunction with Clock Compensation Logic since current calibration pulse is detected in this block. The Clock Compensation Logic detects a calibration pulse of length between 42 and 70 tick counts. This check will assert the CAL_LEN_ERR status bit on failure.

49.4.7.2 Successive calibration pulse check

The successive calibration pulse check checks if successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used. Specifically, the higher the frequency, the closer the check will be to 1.5625%. So if protocol clock is T_{HF_CLK} and channel tick period is T_{TX_CLK} then the check will be of 1.5625% ± {100 × (4 × T_{HF_CLK}) / (42 × T_{TX_CLK})}. So it won't allow variance in clock of a channel by 1.5625% - {100 × (4 × T_{HF_CLK}) / (42 × T_{TX_CLK})}. This check will assert the CAL_DIAG_ERR status bit on failure.

When Option 2 of successive calibration pulse check is enabled, the CAL_RESYNC bit will be asserted on every third successive error detected by this check.

49.4.7.3 Not the expected number of edges check

The number of negative edges between two successive calibration pulses is counted and compared with expected number of edges in case of preferred option of successive calibration pulse check. The expected number of nibbles is the sum of the following:

- Status and Communication nibble (1 No.)
- Data nibbles (Programmable as per user software)
- CRC nibble (1 No.)
- Pause Pulse, if configured (1 No.)

This check ("Not the expected number of edges check") will assert the NUM_EDGES_ERR status bit on failure. The check does not run when Option 2 (low latency) of successive calibration pulse check is enabled. This check is not masked after any other error is detected so it could be redundant if it is flagged along with CAL_DIAG_ERR and CAL_LEN_ERR. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

49.4.7.4 Nibble value check

As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 ticks:

- Status and Communication nibble
- all Data nibbles
- CRC nibble

The nibble value check will assert the NIB_VAL_ERR status bit on failure. This error can be flagged along with NUM_EDGES_ERR in which case the former can be ignored as the number of pulses in the message is incorrect. However, this is true only when CPU is

clearing error status bits as and when they are asserted. In case, errors are accumulated, the above cannot be deduced.

49.4.7.5 CRC check

The CRC check will compute both 4-bit and 6-bit CRC as requested by State Machine control logic. Two 4-bit CRCs and one 6-bit CRC are supported to compute the CRC on Fast/Slow Serial Message (4-bit CRC) and Enhanced Serial Message (6-bit CRC) respectively. The SENT Receiver Module allows the user to calculate the 4-bit CRC using either the legacy CRC implementation method or the new XOR-based CRC implementation via a control bit in the channel's corresponding control register. Refer to SAE SENT Specification for details on CRC algorithm.

Asserting the CHn_CONFIG_REG[11] bit, causes the Status and Communication nibble to be included in the 4-bit CRC calculation for Fast Serial Messages. De-asserting this bit enables normal operation as described above.

This check will assert the FMSG_CRC_ERR or SMSG_CRC_ERR status bit on failure, depending on whether the check is run for Fast Message or Slow Message.

The FMSG_CRC_ERR can be flagged along with the NUM_EDGES_ERR, in which case the CRC error can be ignored as the number of pulses in the message was incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

49.4.7.6 Pause pulse diagnostic

The pause pulse diagnostic checks whether the ratio of calibration pulse length to full message length (including pause pulse) varies from one message to another by less than 1.5625% (as per spec). In order to have reduced latency of operation, message reception is not gated due to this check and no messages are discarded. However, when this check fails, a status bit is asserted to let the software know of this condition. The successive calibration pulse check can be configured to operate along with this check. This configurability is added to avoid conflicting results from the two checks.

There can be some inaccuracy in measuring pause pulse length as SAE specification specifies maximum clock drift/jitter up to maximum nibble length only and the frequency of high frequency receiver clock (protocol clock) used, allows clock compensation logic to ensure accuracy up to maximum nibble length. Hence there could be some inaccuracy in this check when sampling pause pulses that are larger than the maximum nibble pulse length.

This check will assert the PP_DIAG_ERR status bit on failure. This check does not run when reception of pause pulse is disabled.

49.4.8 Time stamp logic

A single counter maintains the time stamp for all channels which can be used by the software to determine the relative time of arrival of the messages for each channel. The time stamp counter is a 32-bit counter.

The clock for this counter is derived from the high frequency receiver clock using a Time Stamp Prescaler Counter that divides the high frequency receiver clock to generate the clock of required resolution. The user must program the time stamp prescaler value (in *Section 49.3.2.1: Global Control Register (GBL_CTRL)*) in order to generate the time stamp clock. The period of the time stamp clock must be less than that of the fastest receiver

channel being used, or it can be set at some fixed value determined by application, such as 1 µs. For example, if the high frequency receiver clock is 60 MHz, the required prescaler value to generate a time stamp clock of 1 µs resolution is 60.

Note: The rollover of time stamp values should be determined in user software, by comparing the time stamp values in previous and current messages that are read out. User software should adjust the time stamp values in these messages accordingly, after reading them.

The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer. However, if the current message is found to be in error, the message buffer is discarded. Thus, the time stamp will be available to user software for only those messages which do not have any error.

The time stamp for a serial message is sampled at the end of the message i.e. when the serial message is completely received and when the message is found to have no errors. the time stamp is stored in the message read buffer to be read by the user software. Again, for messages with error the message and time stamp are discarded.

49.4.8.1 Limitation

Since the same counter is used for all channels, it may happen that more than 1 message is received within the same time stamp across different channels. Hence the software would not be able to determine the sequence of arrival of messages which have the same time stamp value.

49.4.9 Bus Idle Diagnostic

This diagnostic checks the connected sensors for inactivity. SENT module can indicate if the idle period on a particular channel has crossed the period defined by the programmed value in Bus Idle Count (in *Section 49.3.2.20: Channel 'n' Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)*). Status bit (bus_idle in *Section 49.3.2.19: Channel 'n' Status Register (n = 0 to (CH-1)) (CHn_STATUS)*) will be set when idle period crosses the allowed value. CPU should write 1 to this clear this bit once it is read.

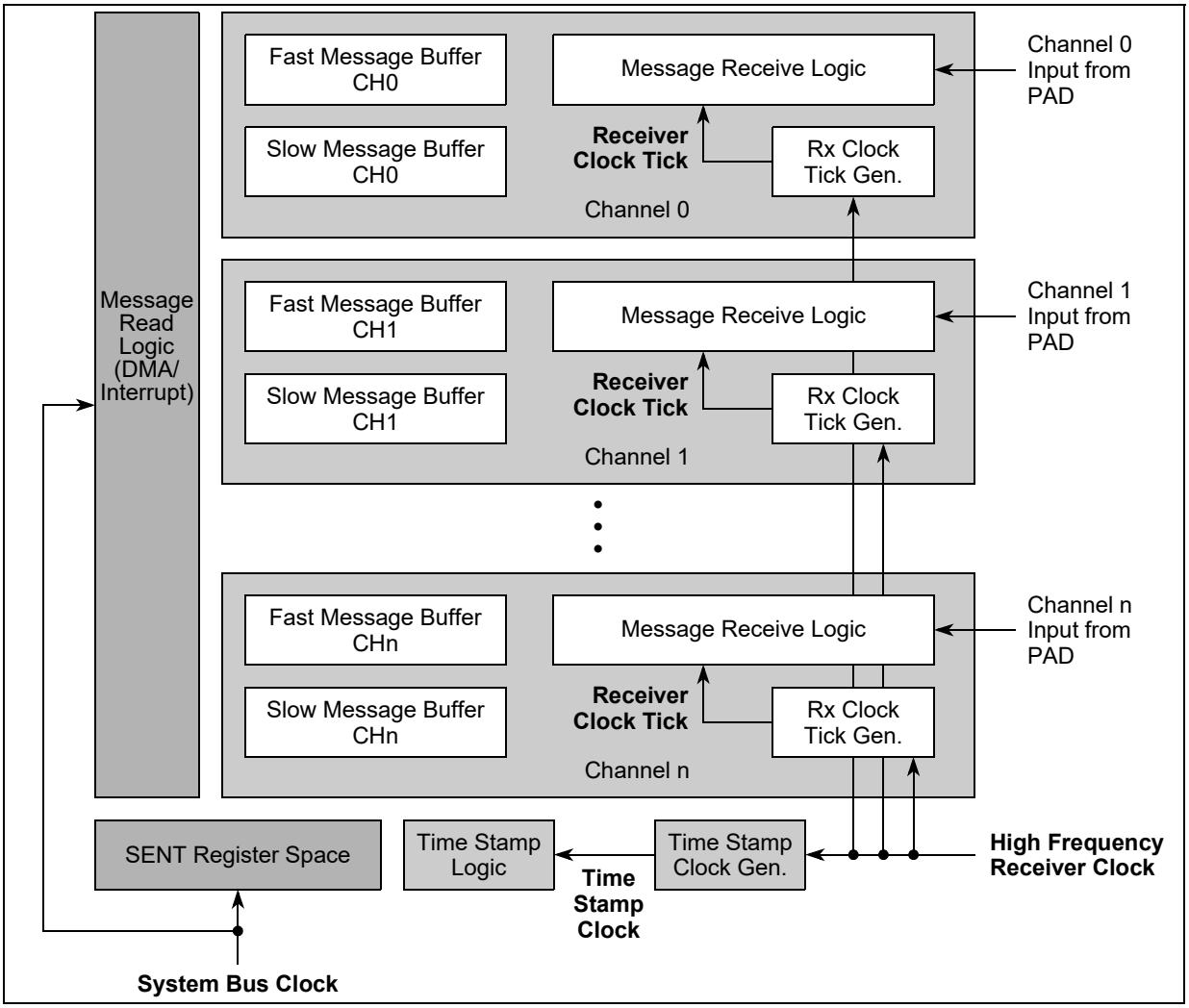
49.5 Clocks and resets

The SENT Receiver works on two clocks. First is the System Bus Clock that is used to program the registers and read messages via DMA or Interrupt from the module's Register Interface. Second is the High Frequency Receiver Clock or Protocol Clock that is used for accurate message receiving operations. There is a single asynchronous reset which is the system asynchronous reset and one internal reset for each of the channel logic.

49.5.1 Clocking strategy

Figure 816 shows the clock domain in which each module is functioning.

Figure 816. SENT Receiver module clock domains



49.5.2 System bus clock requirements

Since all the received messages are read on the System Bus Clock it is necessary that both the interrupt service routine and the DMA controller must be able to access all channels with messages ready to be read within a time to prevent overflow in any channel. The minimum time available to the CPU or DMA is the case when the smallest Fast Message is received back to back on all channels while working on the smallest receive clock tick period.

Now, the smallest Fast Message size with valid checksum as per SAE Specifications is 154 ticks. At 3 µs receive clock tick with a -20% variation in the transmitter clock tick, the effective receiver clock tick is 2.4 µs (after compensation). This gives us the time in which the minimum size Fast Message is received in any channel as 154 $\times$ 2.4 = 369.6 µs. Now since messages are arriving in parallel, 369.6 µs is the time available for CPU or DMA to access all channels to prevent an overflow in the channel that received the very first message.

Thus, if assume that 'N_{CHNL}' channels are being used and all channels have messages ready to be read together from either DMA or CPU then the time available to read a message in one channel will be 369.6/N_{CHNL} µs or ~370/N_{CHNL} µs. Since the CPU or the

DMA needs to make three 32-bit accesses in order to read a complete message and taking the time required for one 32-bit read access as 'T_{RD}' (assuming it to be the same as DMA and CPU); we can determine the minimum frequency for the System Bus Clock as:

For CPU,

Equation 34

$$3 \times T_{RD} + T_{ISR} + T_{WAKUP} < 370/N_{CHNL}$$

where T_{ISR} is the overhead in initiating an ISR and T_{WAKUP} is the time to wakeup from low power mode

For DMA,

Equation 35

$$3 \times T_{RD} + T_{WAKUP} < 370/N_{CHNL}$$

Replacing $T_{RD} = N_{RD} \times T_{SYS_CLK}$ and $T_{ISR} = N_{ISR} \times T_{SYS_CLK}$, where N_{RD} is the number of System Bus Clock cycles in one 32-bit read access, N_{ISR} is the number of System Bus Clock Cycles in initiating an interrupt service routine and T_{SYS_CLK} is the period of the same clock, we get

For CPU,

Equation 36

$$(3 \times N_{RD} + N_{ISR}) \times T_{SYS_CLK} + T_{WAKUP} < 370/N_{CHNL}$$

(all units is µs)

For DMA,

Equation 37

$$3 \times N_{RD} \times T_{SYS_CLK} + T_{WAKUP} < 370/N_{CHNL}$$

(all units is µs)

49.5.3 High frequency receiver clock (protocol clock) requirements

For accurate receive operations, the protocol is required to satisfy a minimum frequency defined as per formula below.

Equation 38

$$F_{HF_CLK} > 332 / (3 \times T_{Tx_CLK})$$

where F_{HF_CLK} is the High Frequency Receiver Clock or Protocol Clock in MHz and T_{Tx_CLK} is the Transmitter or Sensor Clock Frequency in µs.

Note: T_{TX_CLK} in the above equation should be chosen based on the fastest sensor clock among all channels in the device.

50 LINFlexD

50.1 Introduction

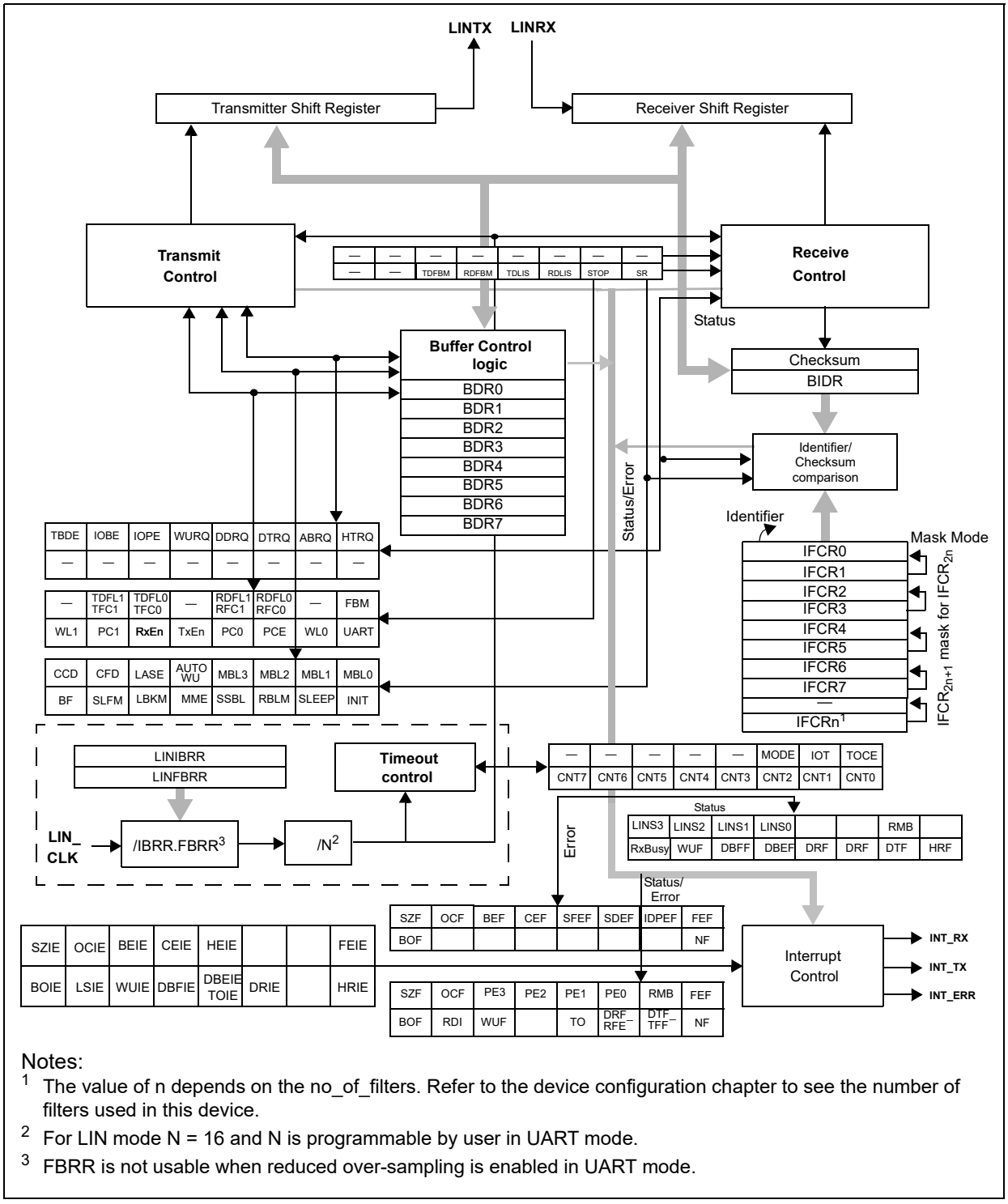
The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

The LINFlexD supports LIN protocol version 1.3, 2.0, and 2.1. 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 817. 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.

50.2 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

50.2.1 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.)

50.2.2 UART mode features

- Full-duplex communication
- Baud rate is a function of baud clock, LINIBRR and LINFBRR registers — see *Section 50.3.2.8, Baud rate generation*
- Separate clock for baud rate calculation (The relationship “(2/3)*LIN_CLK ≥ PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained)
- 15/16/7/8 bits data, parity
- 1/2/3 stop bits
- 12-bit + parity reception
- 4-byte buffer for reception, 4-byte buffer for transmission
- 12-bit counter for timeout management
- 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

50.2.3 Standard features

DMA2x bus interface

50.3 Functional description

50.3.1 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.

50.3.1.1 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 818. Frames

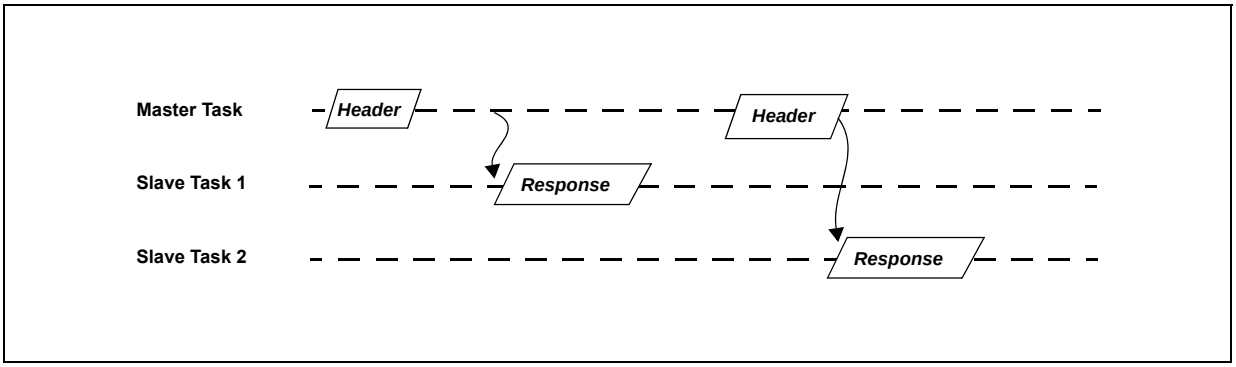
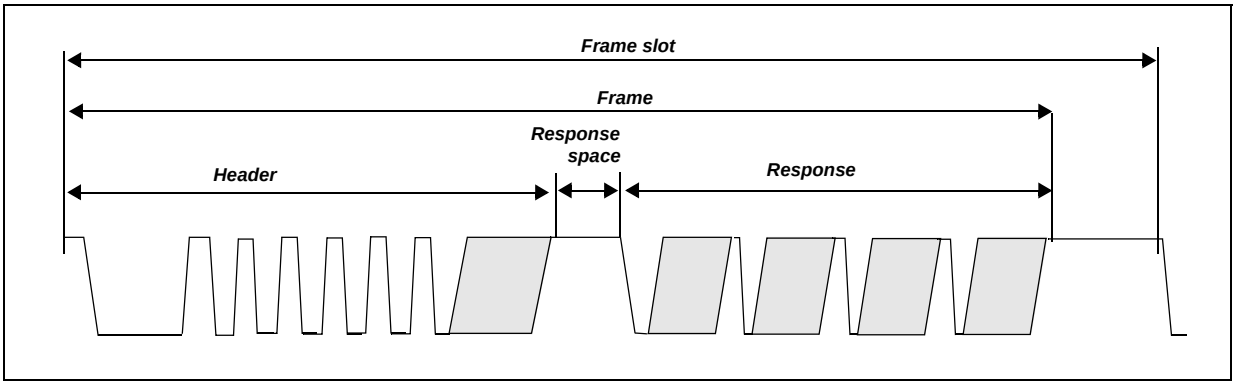


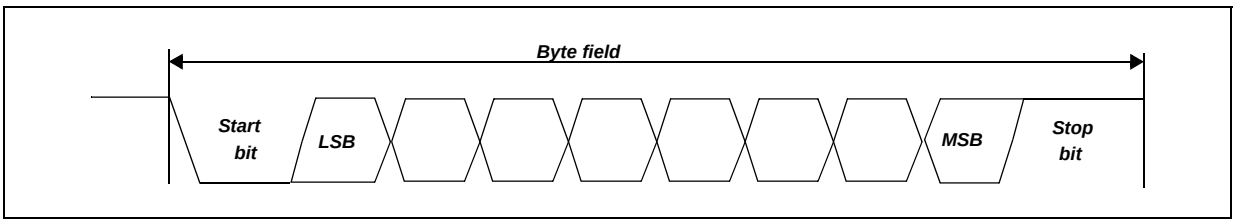
Figure 819. Structure of LIN frame



50.3.1.2 Data field

Each byte is transmitted as shown in *Figure 820*. 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).

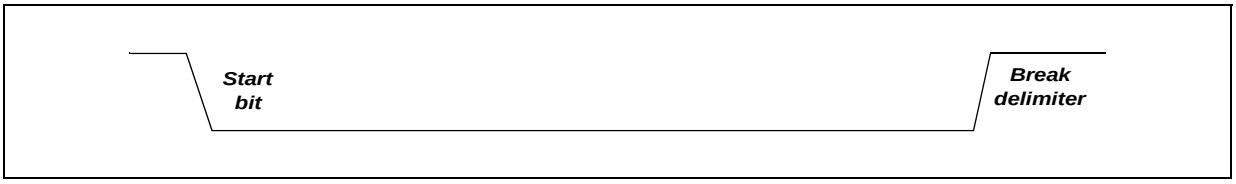
Figure 820. Structure of byte field



50.3.1.3 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 821*. The break delimiter must be of two-bit duration (to be compliant with LIN protocol 2.1).

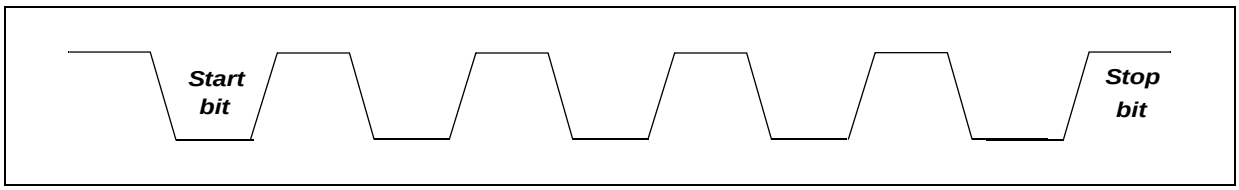
Figure 821. Break field



50.3.1.4 Sync byte

Sync is a byte field with the data value of 0x55.

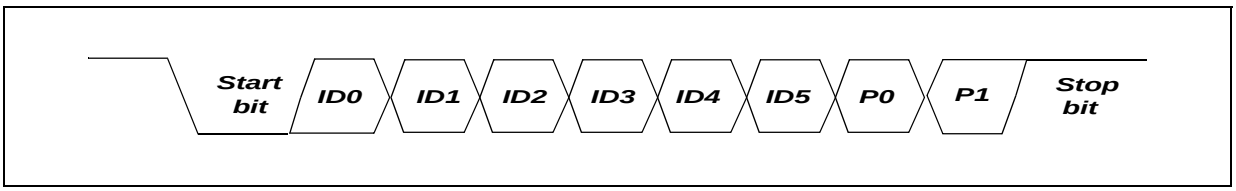
Figure 822. Sync byte field



50.3.1.5 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 823. Identifier field



P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

50.3.1.6 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.

50.3.2 LINFlexD features

50.3.2.1 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.

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)

Normal mode (NM)

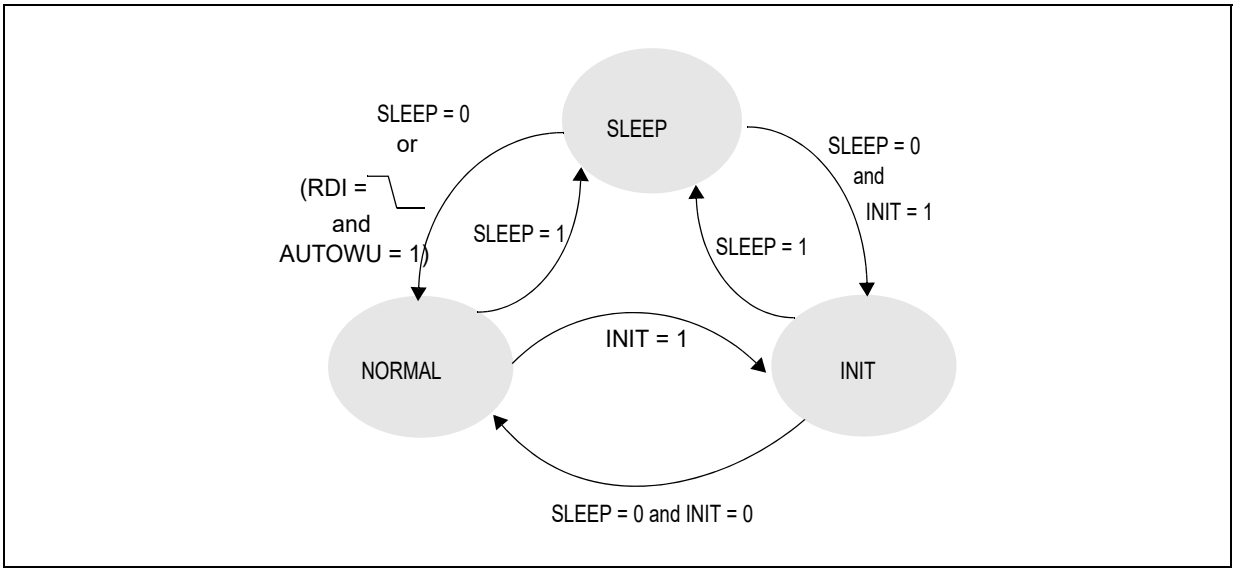
Once software has completed initialization of the LINFlexD controller, it can enter Normal mode by clearing the INIT bit.

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 50.3.2.10, Wakeup management*.

Figure 824. Operating modes



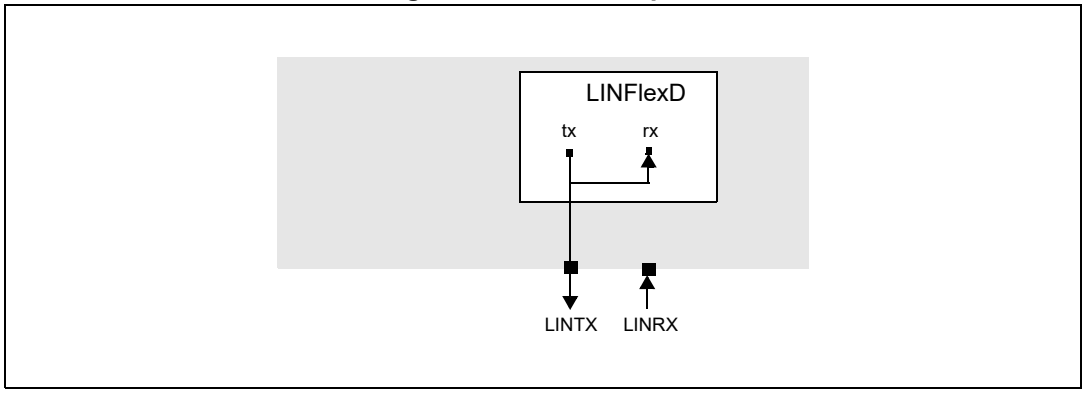
50.3.2.2 Test mode

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 825. LinFlexD pins



50.3.2.3 Master mode

Master mode is selected by means of the MME bit in LINCR1 register.

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.

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.

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.

Data Discard

If the user wants to discard the data after header transmission, then the DDRQ bit in LINCR2 should be set.

50.3.2.4 Slave mode

This mode is selected when the MME bit of the LINCR1 register is cleared.

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.

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.

50.3.2.5 Errors

50.3.2.5.1 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.

3. 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.

50.3.2.5.2 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 21.

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.

Note: 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.

50.3.2.5.3 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.

50.3.2.5.4 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.

50.3.2.5.5 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.

50.3.2.5.6 Timeout error

Figure 826. Incomplete response (for example, missing checksum)

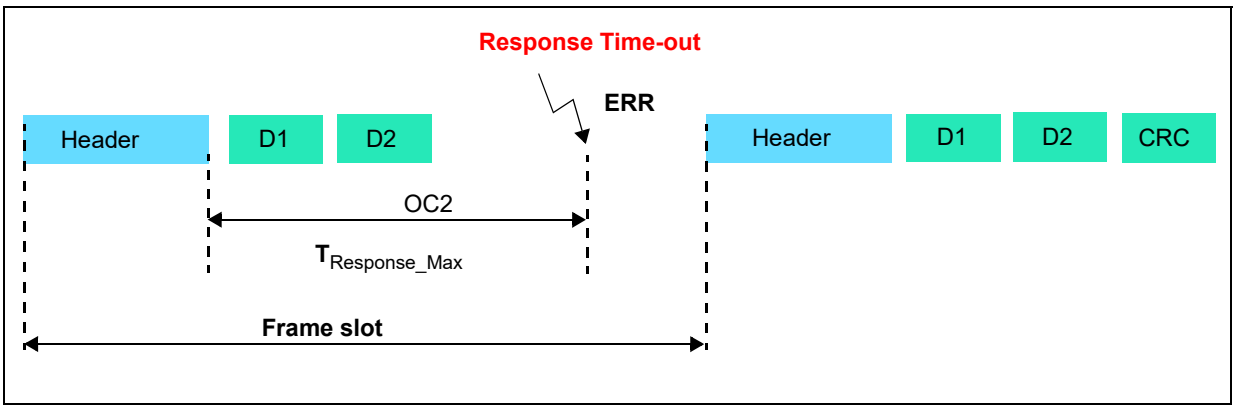
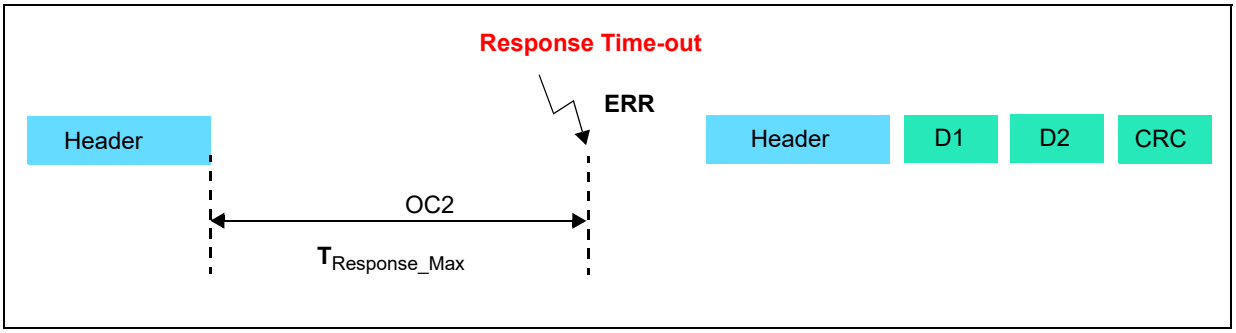


Figure 827. No response



Response timeout mechanism

- Master mode
  - OC2 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 OC2 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 OC2 value (no change compared to earlier implementation).
  - In case of an incomplete response, timeout also occurs when the counter reaches the OC2 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, OC2 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, OC2 is loaded with 36 (maximum possible response space) + LINFlexD_LINTCSR[CNT]
    - 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 not 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).

50.3.2.5.7 Noise

During reception each bit is sampled 16 times and the value of the bit is obtained by taking the majority value of the 8^{th}, 9^{th}, and 10^{th} 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 4^{th}, 5^{th}, and 6^{th} 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 50.4.2.4, LIN Error Status Register (LINESR)*).

50.3.2.6 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.

Identifier list mode

If the n^{th} bit of IFMR is cleared, then filter number 2*n* and 2*n*+1 is in identifier list mode. In this mode, the maximum number of filters that can be configured for transmission/reception equals no_of_filters, depending on the FACT bit of IFER. In this mode the identifier received should match bit by bit to the ID field of IFCR2*n* or IFCR2*n*+1 (if the corresponding FACT bit is set).

Identifier mask mode

If the number of filters required is more than no_of_filters, then filters should be configured in Mask mode. If the n^{th} bit of IFMR is set, then filter number 2*n* is the filter and 2*n*+1 acts as a mask for it. The FACT bit for filter 2*n*+1 has no effect in this case. In this mode, if the x^{th} bit of the mask is set, then the x^{th} bit of the received identifier must match the x^{th} bit of filter.

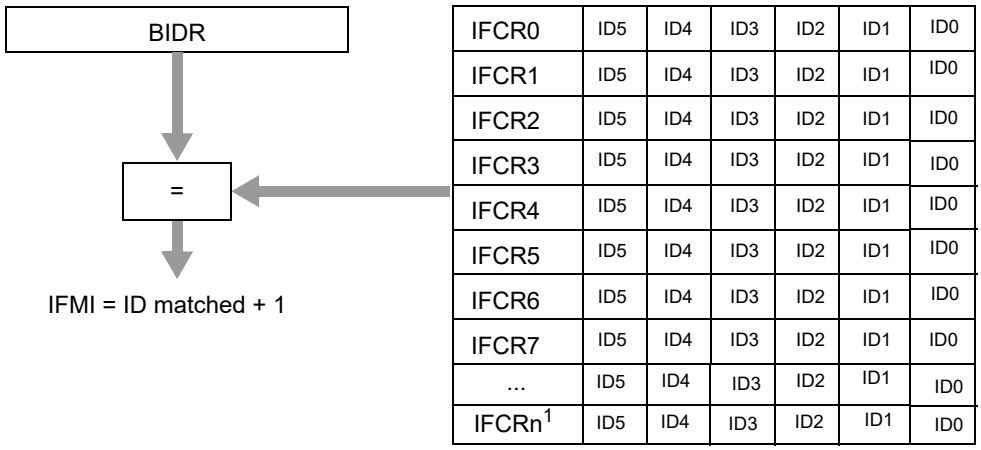
If there is a match of identifier with the *m*th filter in any mode, then *m*+1 is loaded in the IFMI register by hardware. No match condition is denoted by IFMI = 0.

Upon matching DFL (2:0), the CCS and DIR bits of the BIDR register are copied from the filter by hardware and from then on, the BIDR register is read-only until the end of the frame. Now if the DIR bit of BIDR is set, then a TXI interrupt is generated if the HRIE bit of LINIER is set. In this case, software uses the IFMI register to transfer the relevant data from the RAM area to BDR, and after a complete transfer the DTRQ bit of LINCR2 is set to start the transmission. If the DIR bit is cleared then an RXI interrupt is generated (provided DRIE bit of LINIER is set) when the checksum has been received and there is no checksum error.

In case of no filter match condition (IFMI = 0) and if the BF bit of LINCR1 is set, then an RXI is generated. Now it is the responsibility of software to configure BIDR and start transmission (by loading the BDR buffer and setting the DTRQ bit of LINCR2) or discard reception (by setting the DDRQ bit of LINCR2). If the BF bit is reset, then the receiver discards the received identifier and turns to Idle state in search of a new break.

Note: If one identifier matches with two filters (one is in the list and the other in Mask mode) then List mode prevails over Mask mode. In Mask mode, if two filters match with the identifier then the filter having the lower number prevails.

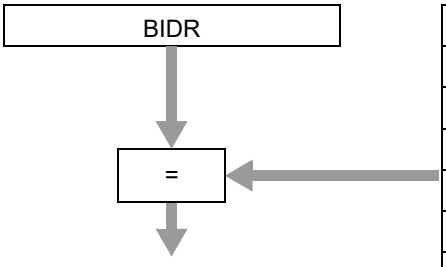
Figure 828. Identifier List mode



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 829. 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.

50.3.2.7 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 830. 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.

50.3.2.7.1 Start detection mechanism

The following steps are followed for detecting a start bit:

1. Refer to *Table 830* 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 (i.e., 2) out of these 3 samples are equal to ‘0’, 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.

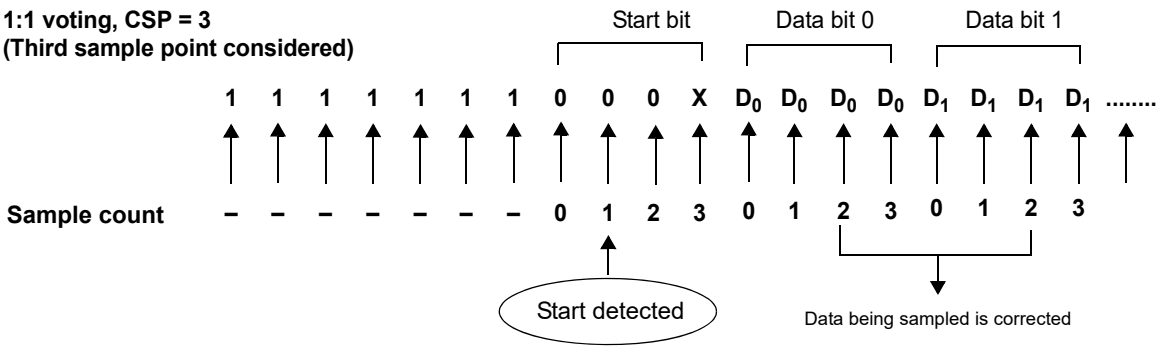
50.3.2.7.2 Delimiter detection mechanism

The following steps are followed for detecting a delimiter bit:

1. Refer to *Table 830* 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 (i.e., 2) out of these 3 samples are equal to ‘1’, 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.

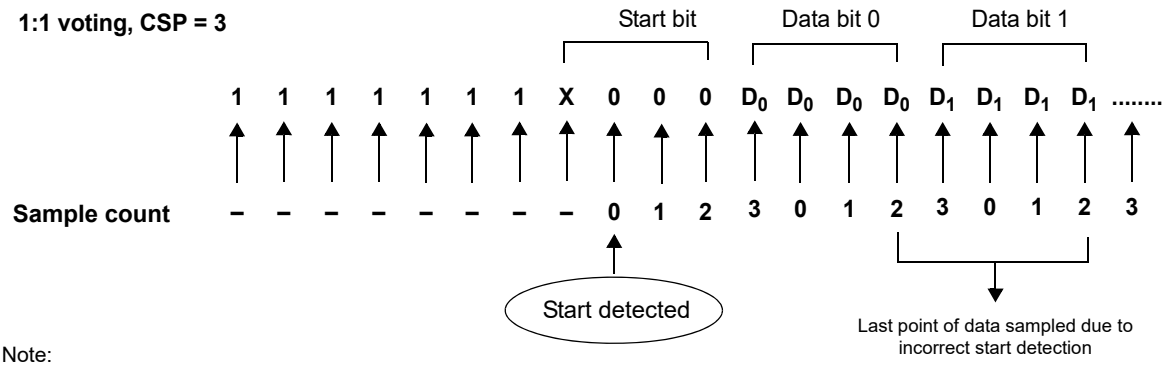
Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two out of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

Figure 830. 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 831. Start detection and sampling for over sampling rate = 4 (Case 2a)

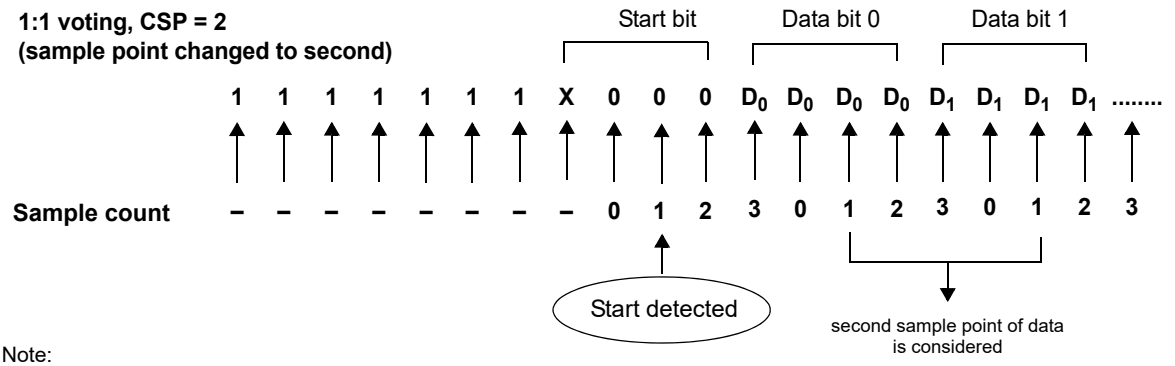


Note:
1) D_0, D_1: Databits
2) X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a indeterministic value.

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.

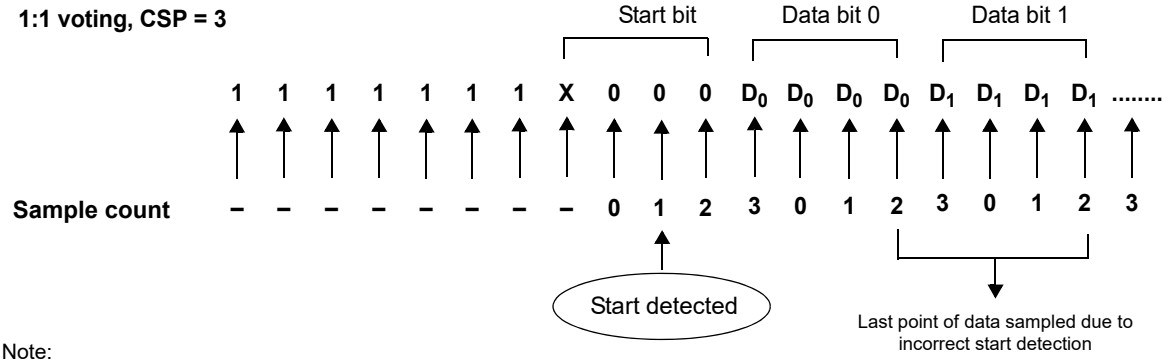
To sample at the correct point therefore, CSP has to be changed to two; resulting in the following situation:

Figure 832. Start detection and sampling for over sampling rate = 4 (Case 2b)



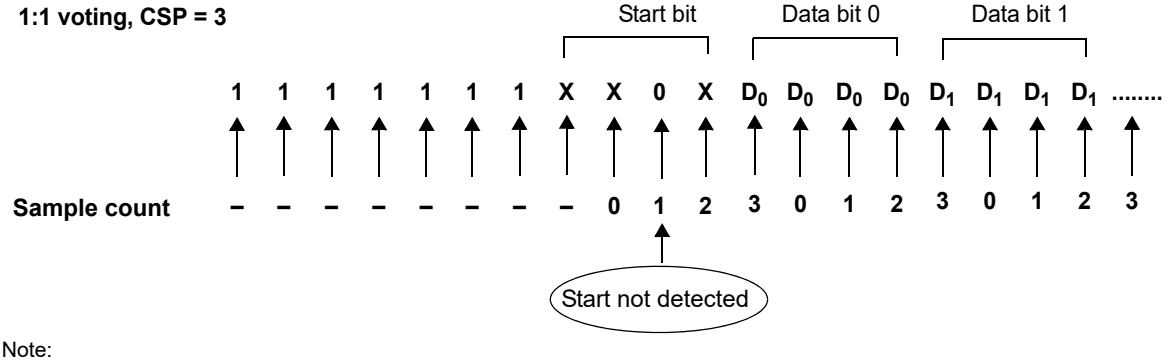
Note:
1) D_0, D_1: Databits
2) X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a non deterministic value.

Figure 833. Start detection and sampling for over sampling rate = 4 (Case 3)



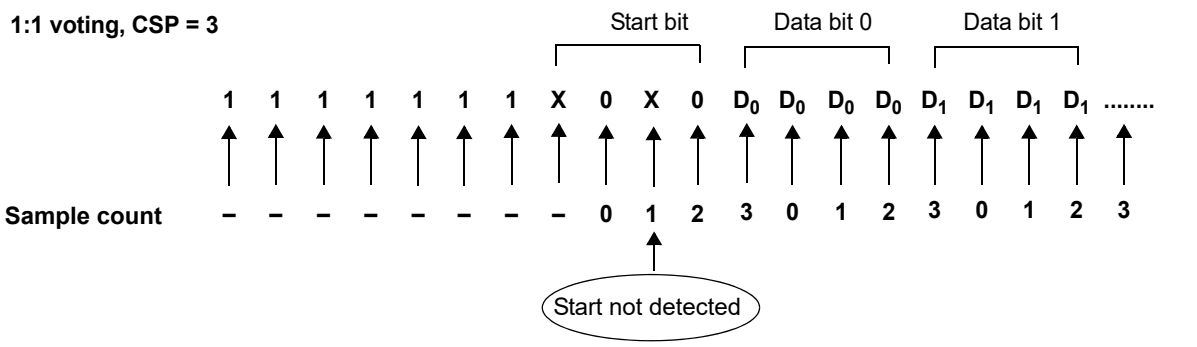
Note:
1) D_0, D_1: Databits
2) X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a non deterministic value.

Figure 834. 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. Therefore ‘X’ refers to a non deterministic value.

Figure 835. 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. Therefore 'X' refers to a non deterministic value.

50.3.2.8 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 39
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 22.

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

Note: Due to design constraints the maximum baud rate achievable is 25 Mbit/s.

*Note: The relationship "(2/3)*LIN_CLK ≥ PBRIDGEx_CLK > 1/3*LIN_CLK" should also be maintained.*

50.3.2.9 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.

50.3.2.10 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.

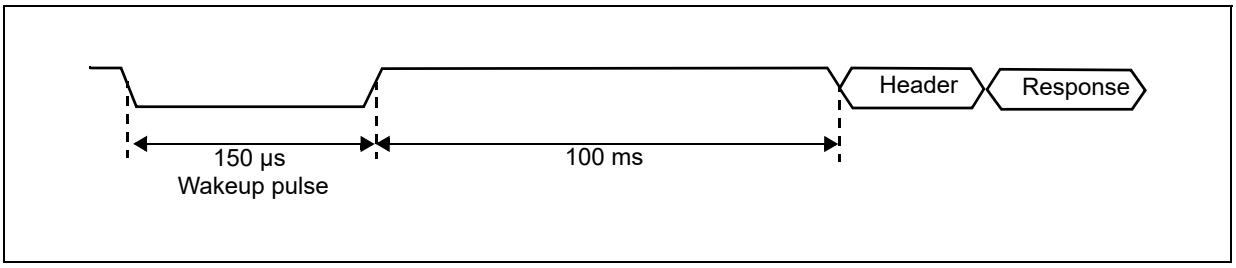
1. AUTOWU = 1

   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 836. Wakeup sequence



50.3.3 Timer

There is an 8-bit counter which has different behavior in different modes.

In Output Compare Mode (LINTCSR[MODE]= 1) : This counter is running even in Sleep and Initialization mode. The counter value which when matches the two software configurable output compare registers (LINOCR[OC1] or LINOCR[OC2]), generate output compare interrupt (LINESR[OCF]) provided TOCE bit in LINTCSR is set. Once an interrupt occurs subsequent interrupt is generated only when the application has soft reset the LIN IP through GCR [SR] bit to get reset the output compare flag (LINESR[OCF]).

In LIN Mode : The software has no control over the TOCE bit and output compare registers are utilized for generation of LIN timeout interrupts (header, frame, response times out). In this case, if LIN moves to Sleep or Init state then this counter remains in Reset state. LIN mode has no meaning if UART is enabled, hence the counter will remain in Reset state.

50.3.4 UART mode

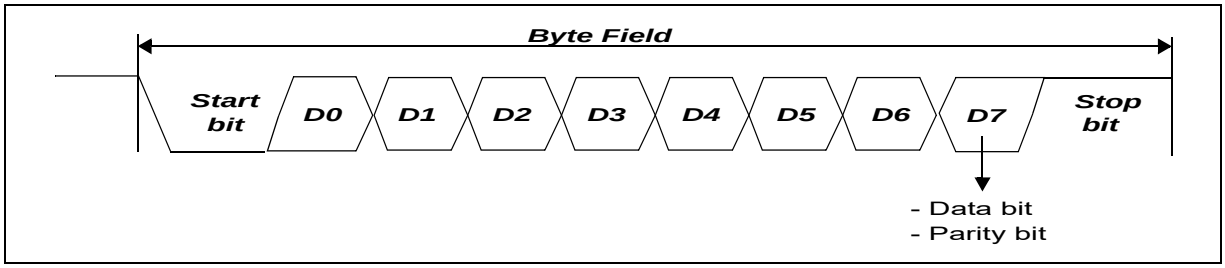
The main features in UART mode are:

- 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

50.3.4.1 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.

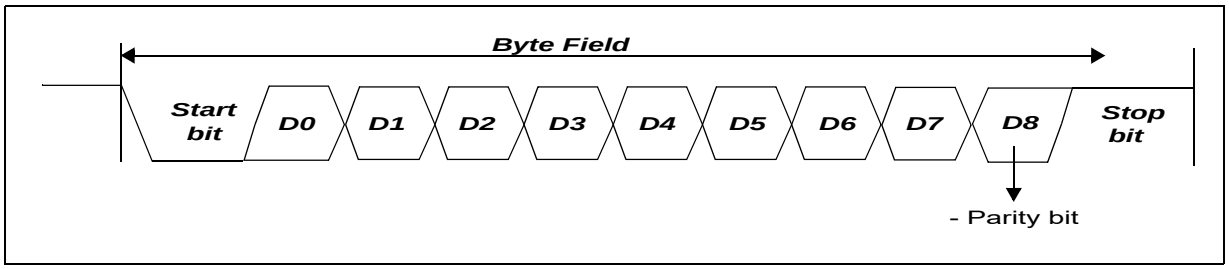
Figure 837. UART mode 8-bit data frame



50.3.4.2 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.

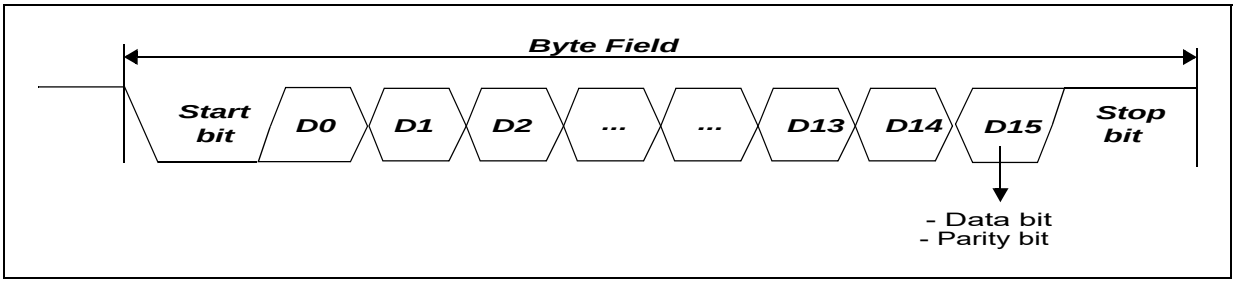
Figure 838. UART mode 9-bit data frame



50.3.4.3 16-bit data frames

The sixteenth 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.

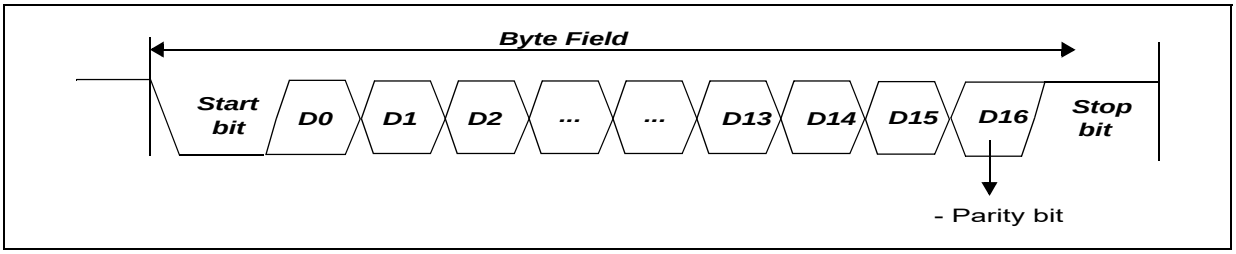
Figure 839. UART mode 16-bit data frame



50.3.4.4 17-bit frames

The seventeenth 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.

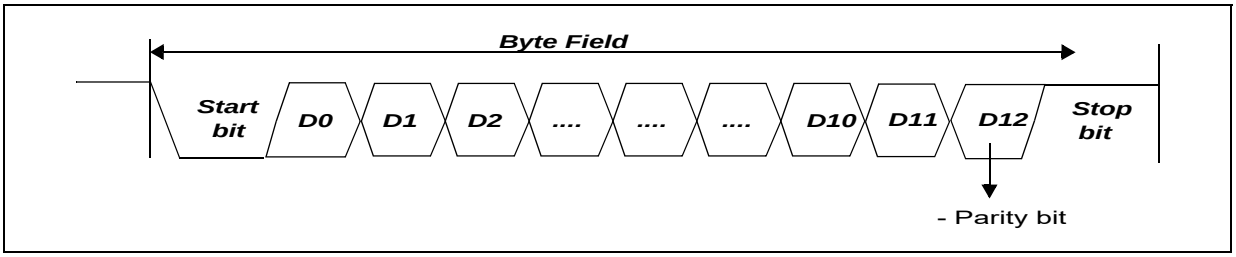
Figure 840. UART mode 17-bit data frame



50.3.4.5 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.

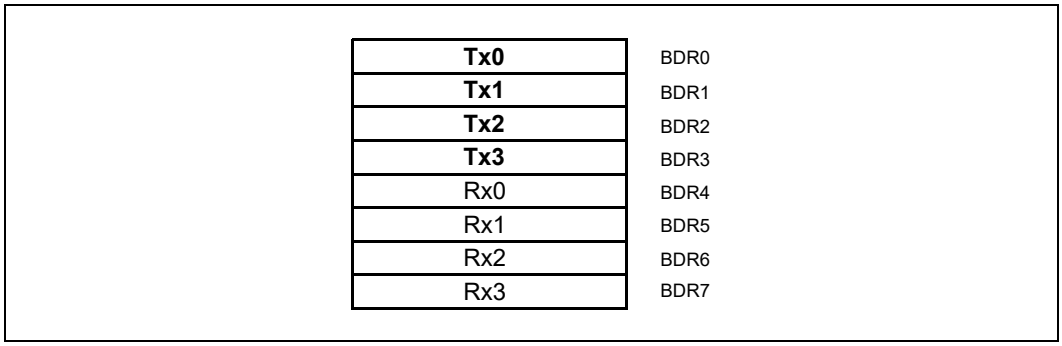
Figure 841. UART mode 13-bit data frame



50.3.4.6 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter, as shown in *Figure 842*.

Figure 842. 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.

50.3.4.7 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.

Note: Invalid writes to BDRL which generates IPS transfer error, also triggers data transmission at the same time.

Note: 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 831. BDRL access in UART mode

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR:WL)(1)	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 831. BDRL access in UART mode(Continued)

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR:WL)(1)	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.

50.3.4.8 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 832. BDRM access in UART mode

IPS operation	Register(1)	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)(2)	IPS operation result
Read byte4	BDRM	FIFO	Byte	OK
Read byte5-6-7	BDRM	FIFO	Byte	IPS transfer error
Read halfword2-3	BDRM	FIFO	Byte	IPS transfer error
Read word	BDRM	FIFO	Byte	IPS transfer error
Read byte4-5-6-7	BDRM	FIFO	Halfword	IPS transfer error
Read halfword2	BDRM	FIFO	Halfword	OK
Read halfword3	BDRM	FIFO	Halfword	IPS transfer error
Read word	BDRM	FIFO	Halfword	IPS transfer error

Table 832. BDRM access in UART mode(Continued)

IPS operation	Register(1)	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)(2)	IPS operation result
Write byte4-5-6-7	BDRM	FIFO	Byte/halfword	IPS transfer error
Write halfword2-3	BDRM	FIFO	Byte/halfword	IPS transfer error
Write word	BDRM	FIFO	Byte/halfword	IPS transfer error
Read byte4-5-6-7	BDRM	Buffer	Byte/halfword	OK
Read halfword2-3	BDRM	Buffer	Byte/halfword	OK
Read word	BDRM	Buffer	Byte/halfword	OK
Write byte4-5-6-7	BDRM	Buffer	Byte/halfword	IPS transfer error
Write halfword2-3	BDRM	Buffer	Byte/halfword	IPS transfer error
Write word	BDRM	Buffer	Byte/halfword	IPS transfer error

1. Refer to the layout of the registers BDRL and BDRM to identify the mapping between byte x and the data bits of the BDRL and BDRM registers.
2. Halfword0 refers to DATA1 and DATA0 of BDRL register. Halfword1 refers to DATA3 and DATA2 of BDRL register. Halfword2 refers to DATA5 and DATA4 of BDRM register. Halfword3 refers to DATA7 and DATA6 of BDRM register.

- *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.*
- *When IPG_STOP is requested, for UART in FIFO mode, SW has to set the INIT bit in order to send the receiver to idle. This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.*

50.3.5 DMA interface

50.3.5.1 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.

50.3.5.2 Definitions

Control/status register fields of the LINFlexD macrocell are described in *Table 833* and *Table 834*.

Table 833. 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 833. 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 834. 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 TCD descriptors referred to in this document are described in *Table 835*.

Table 835. 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

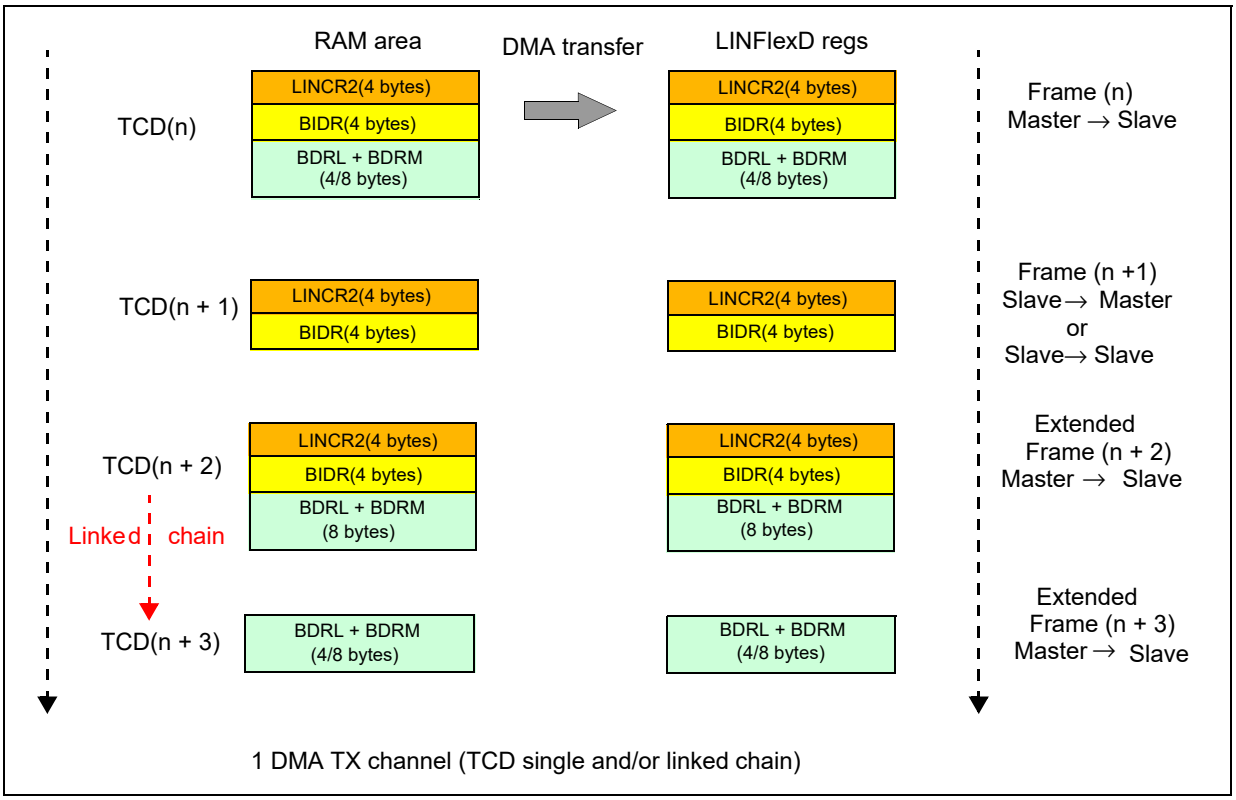
Table 835. TCD control fields description(Continued)

TCD field	Level	Description
DLAST_SGA[31:0]	—	Last destination address adjustment or the memory address for the next TCD to be loaded into this channel (scatter/gather)
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.

50.3.5.3 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 843*.

Figure 843. 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 836 provides the register settings of the LINCR2 and BIDR for each class of LIN frame.

Table 836. Master node — Tx mode — Register setting

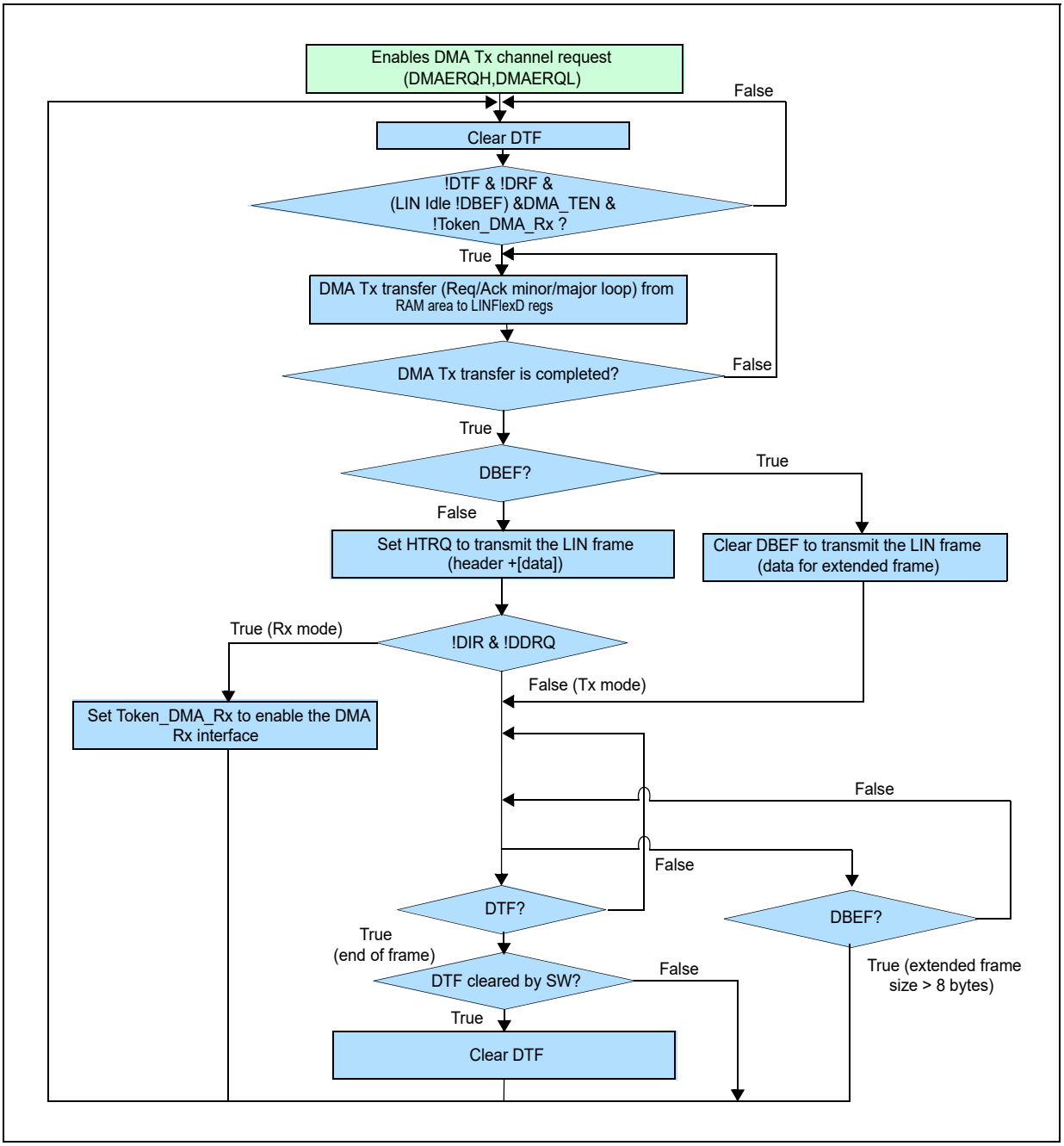
LIN frame	LINCR2	BIDR
Master to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)
Slave to Master	DDRQ = 0 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)
Slave to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Tx interface is given in *Figure 844*. 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 837*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 837. 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

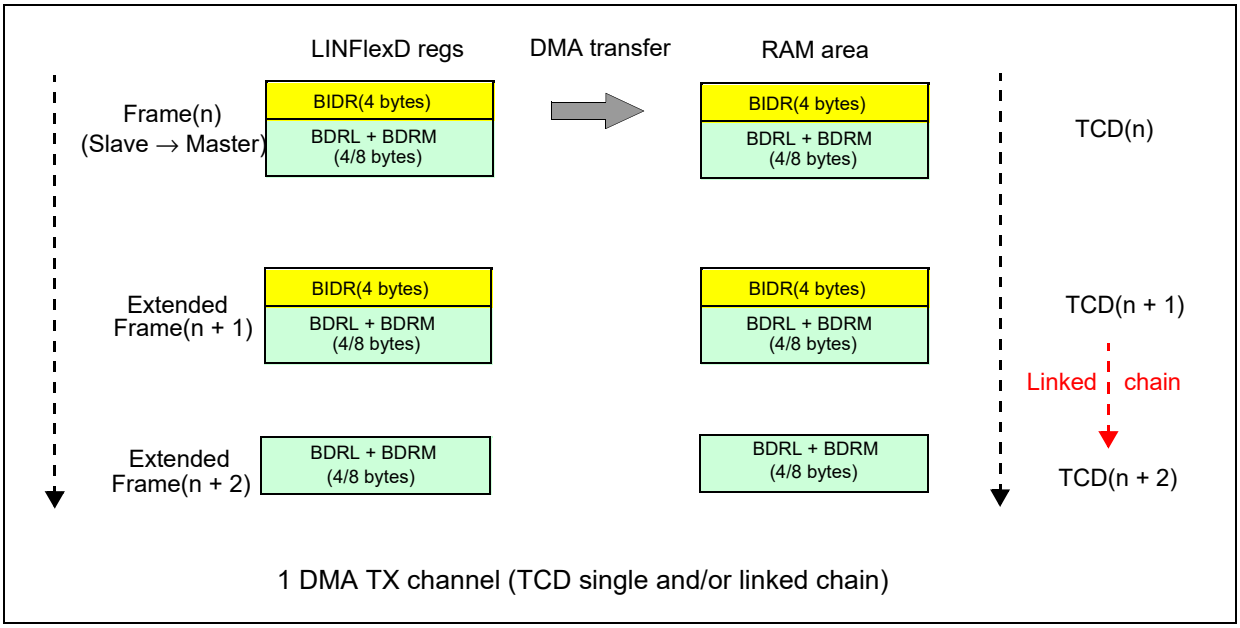
Figure 844. Master node — DMA Tx FSM (concept scheme)



50.3.5.4 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 845*.

Figure 845. 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 846*. 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 838*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

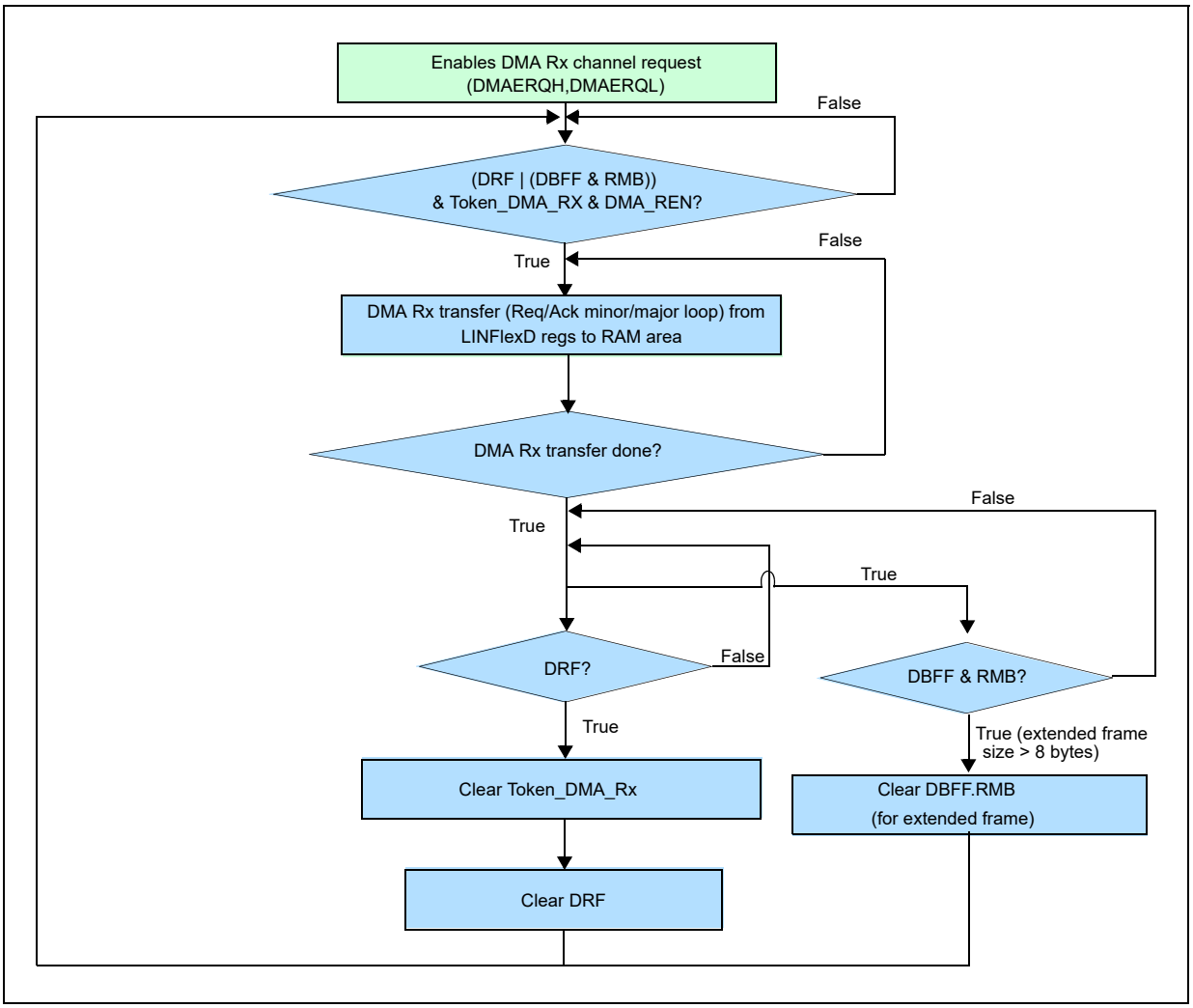
Table 838. TCD setting — Master node — Rx mode

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 838. 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

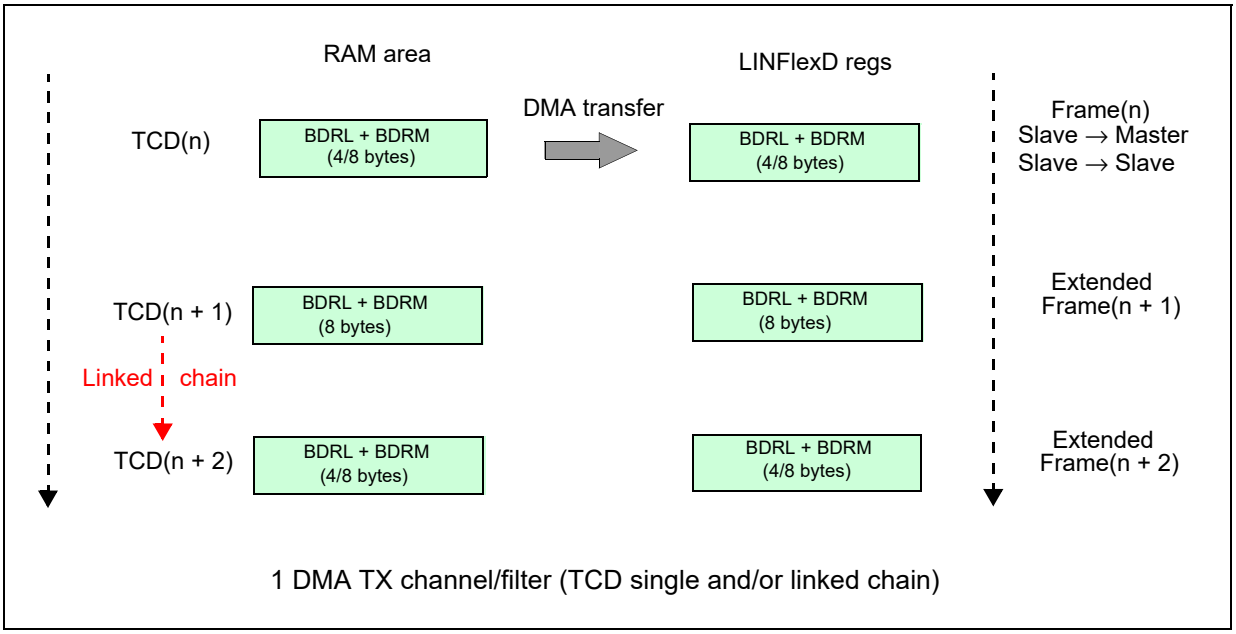
Figure 846. Master node — DMA Rx FSM (concept scheme)



50.3.5.5 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 847*.

Figure 847. 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 839*.

Table 839. 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 848*. DMA TX FSM moves to Idle state if DMATXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in *Table 840*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

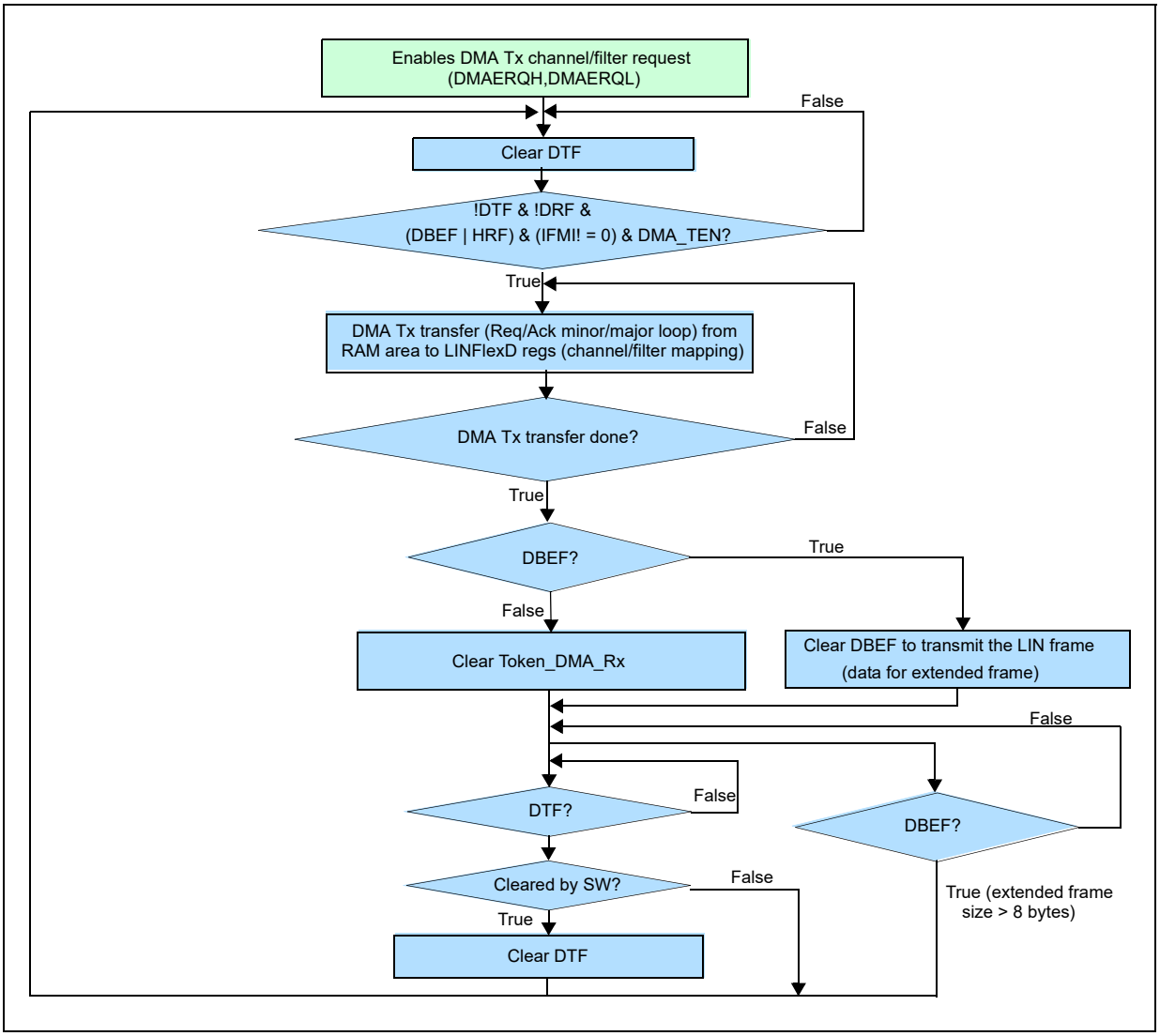
Table 840. TCD setting — Slave node — Tx mode

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 840. 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

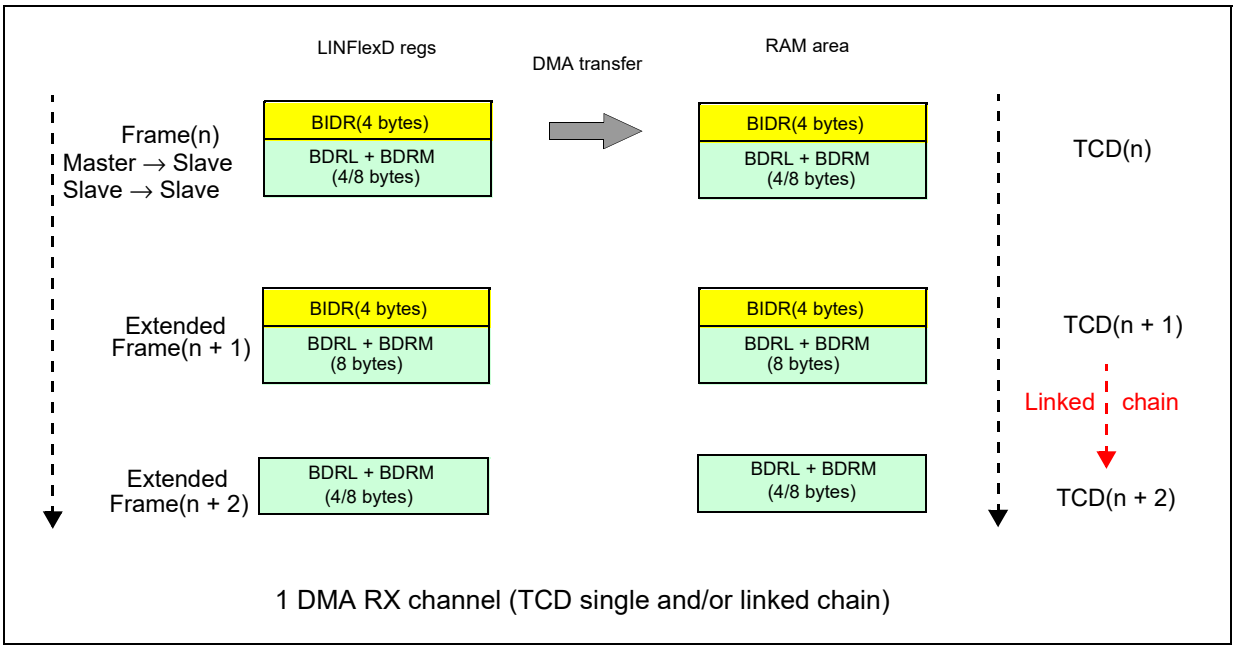
Figure 848. Slave node — DMA Tx FSM (concept scheme)



50.3.5.6 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 849*.

Figure 849. 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 841*.

Table 841. Slave node – Rx mode — Register setting

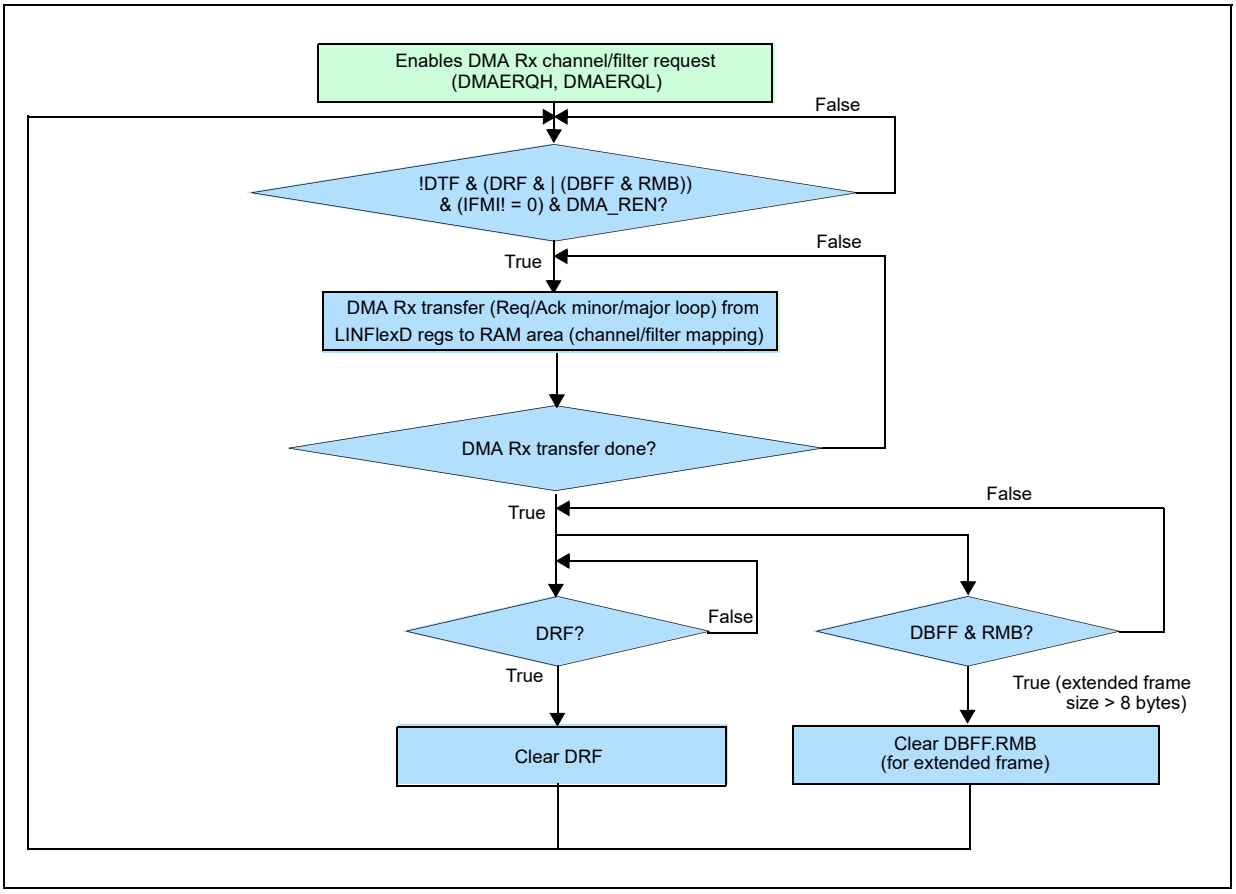
LIN frame	LINCR2	IFER	IFMR	IFCR
Master to Slave or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Rx mode) for each DMA RX channel	– Identifier list mode – Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Rx interface is given in *Figure 850*. DMA Rx FSM moves to Idle state if DMARXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in *Table 842*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 842. 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

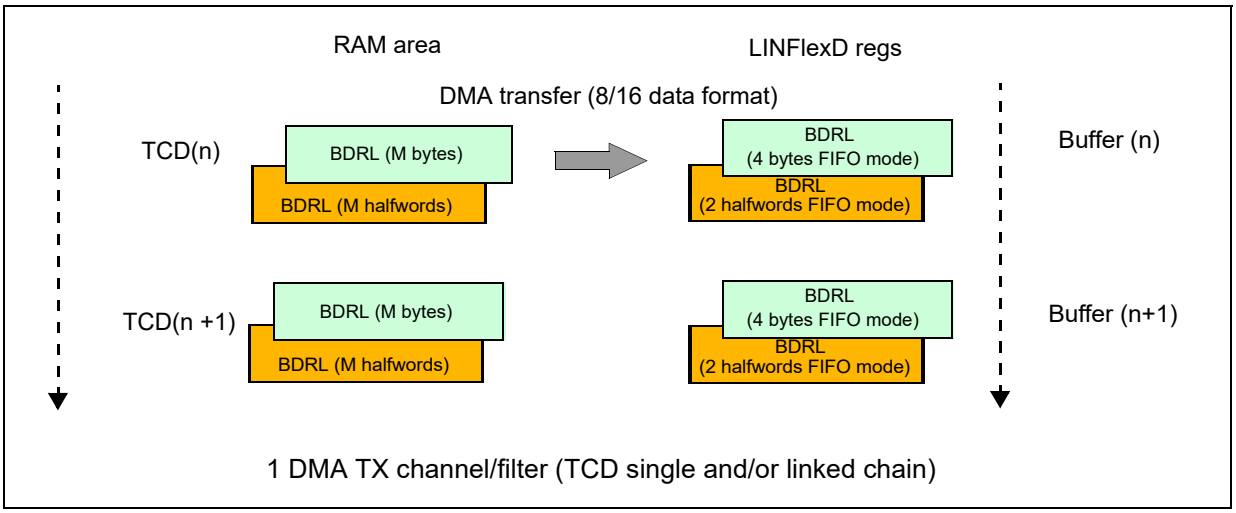
Figure 850. Slave node — DMA Rx FSM (concept scheme)



50.3.5.7 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 851*.

Figure 851. 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.

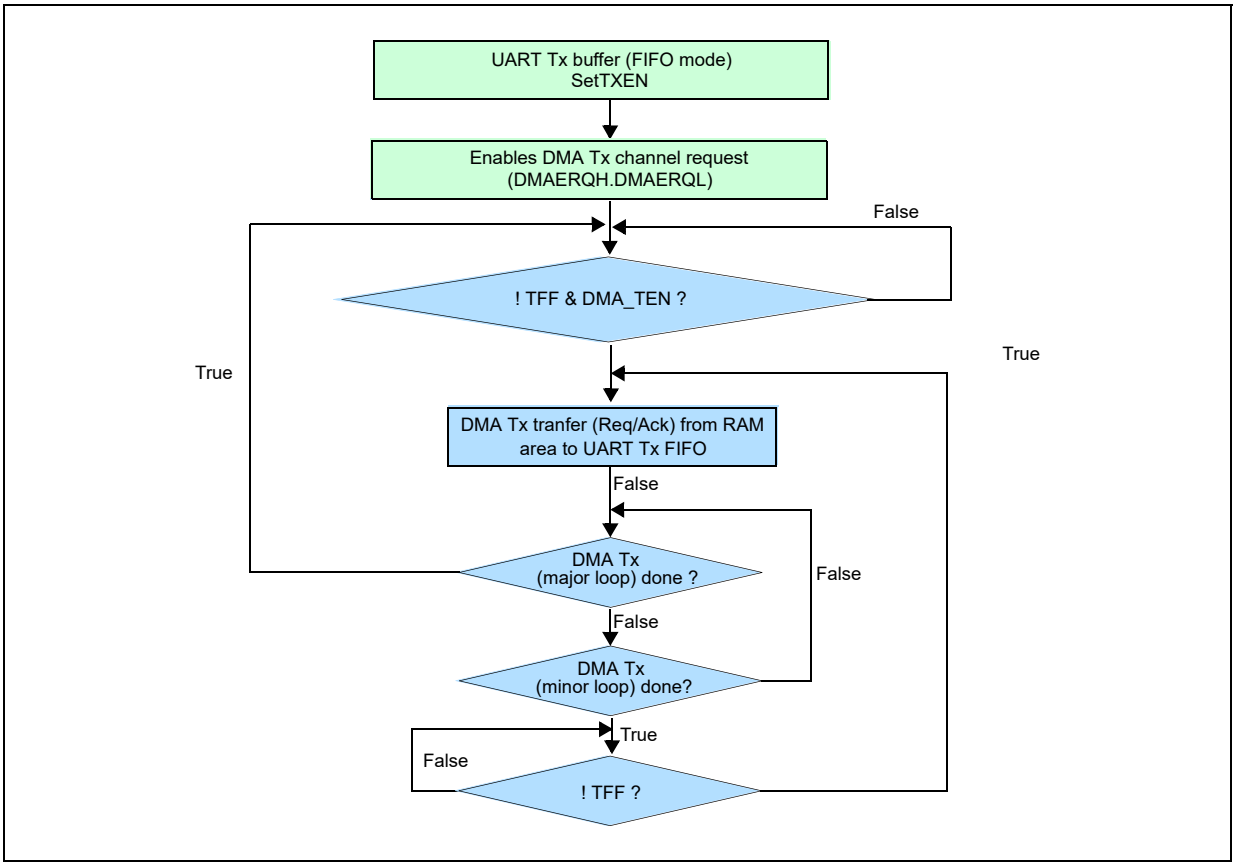
The concept FSM to control the DMA Tx interface is given in *Figure 852*. DMA Tx FSM will move to Idle state if DMATXE[0] = 0. The TCD setting (typical case) is given in *Table 843*. All other TCD fields = 0. The minor loop transfers a single byte/halfword as soon as a free entry is available in the Tx FIFO.

Table 843. 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.

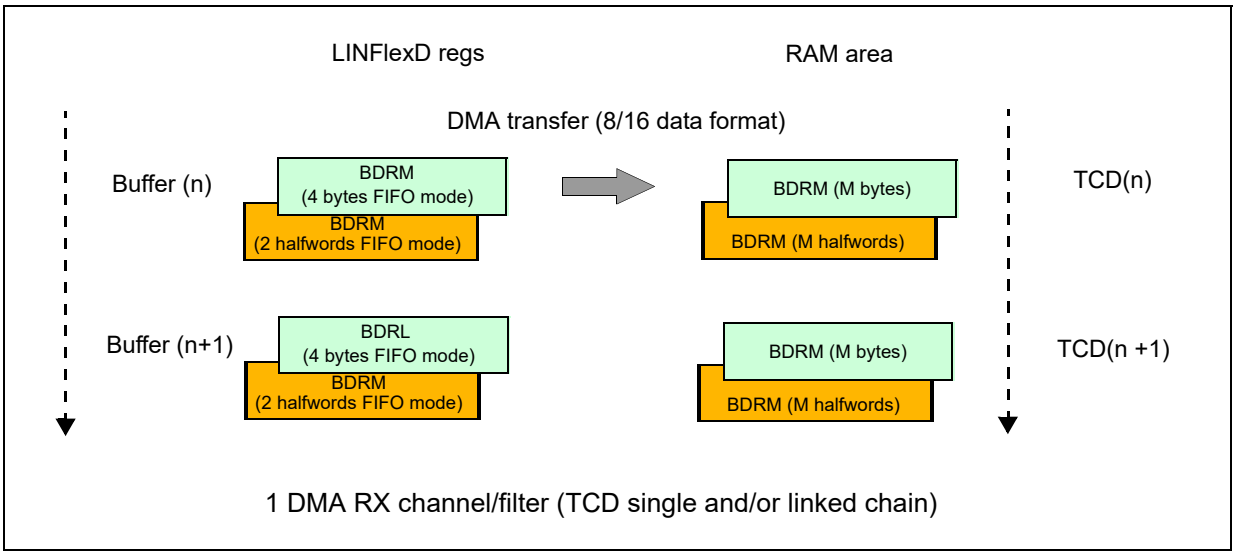
Figure 852. UART — DMA Tx FSM (concept scheme)



50.3.5.8 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 844*.

Figure 853. 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.

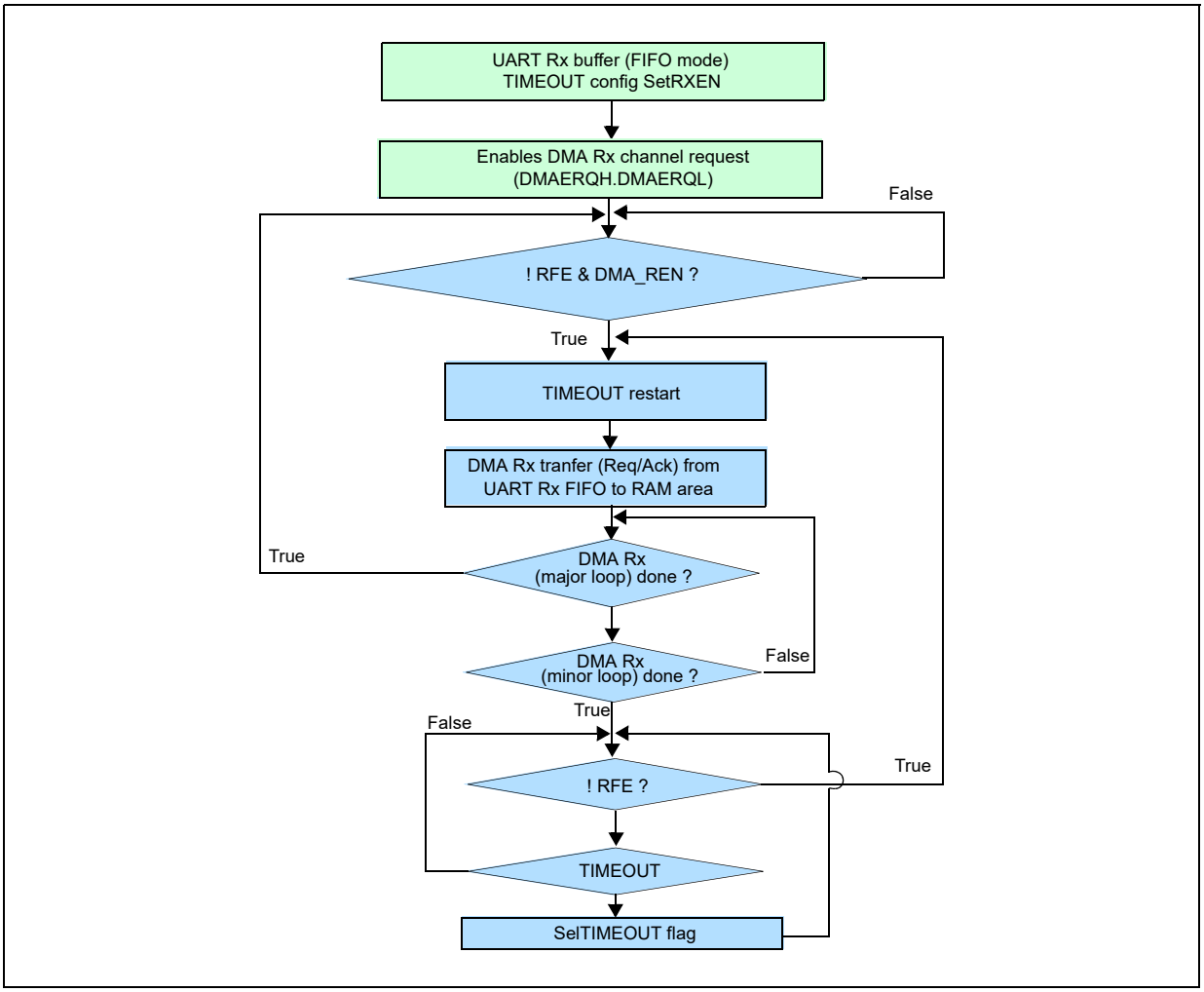
The concept FSM to control the DMA Rx interface is given in *Figure 854*. DMA Rx FSM will move to Idle state if DMARXE[0] = 0. The TCD setting (typical case) is given in the next table. All other TCD fields equal zero. The minor loop transfers a single byte/halfword as soon an entry is available in the 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 844. 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 854. UART — DMA Rx FSM (concept scheme)



50.3.5.9 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 855. DMA interface (major loop — single iteration)

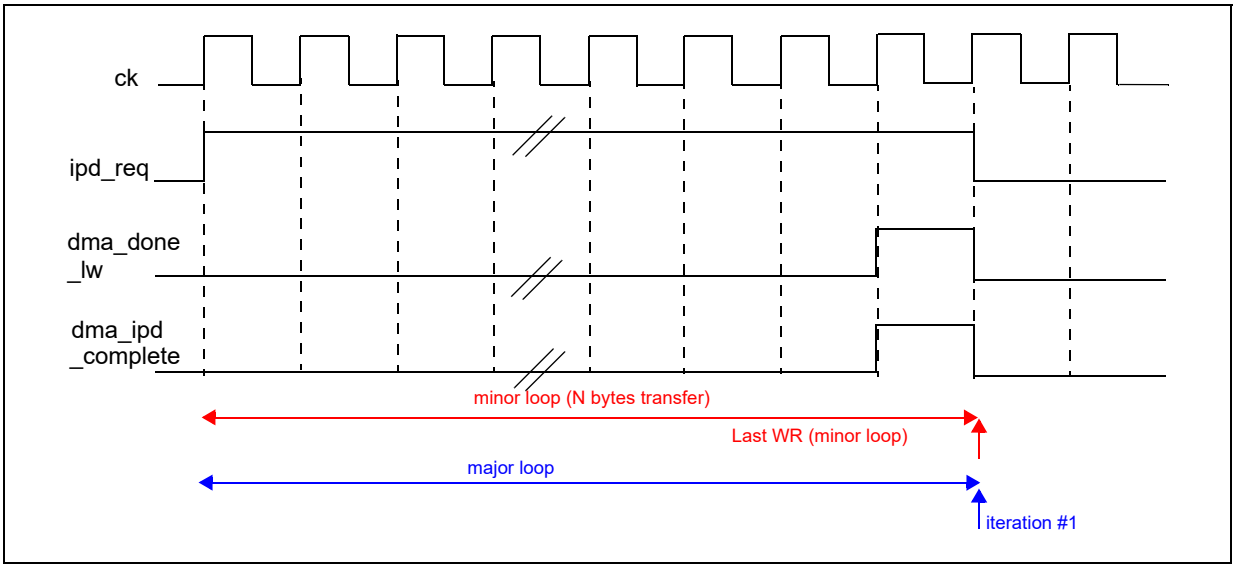
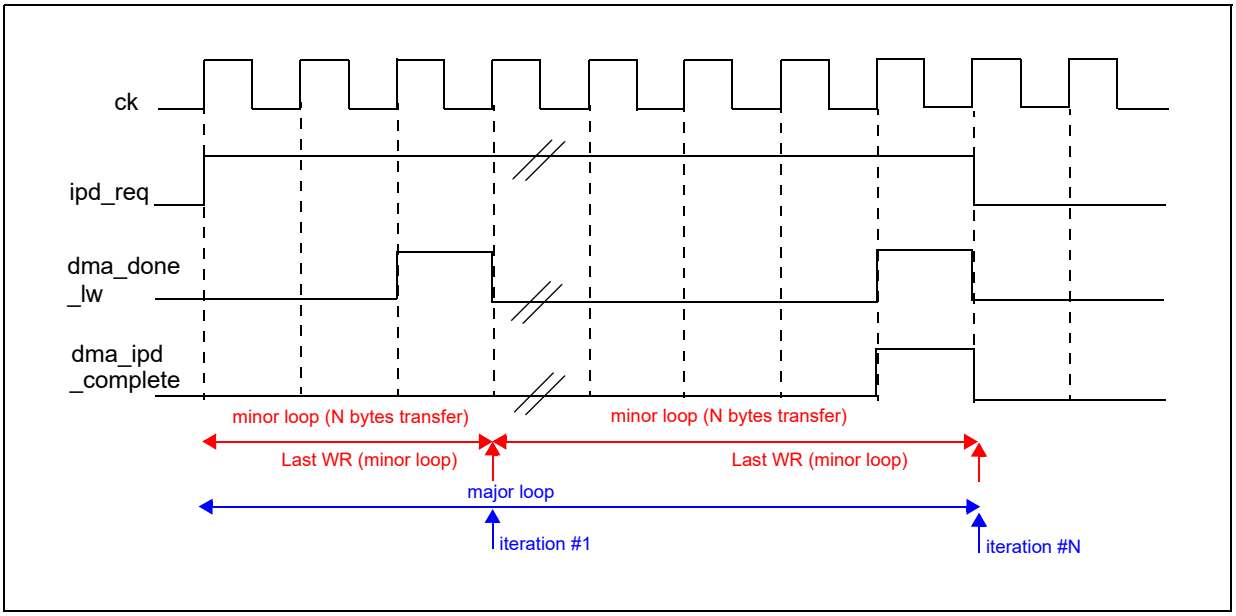


Figure 856. DMA interface (major loop — N iteration)



50.3.5.10 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 cancelling 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.

50.4 Memory map and register description

50.4.1 Memory map

Table 845 shows the LINFlexD memory/register map.

Note: In Master mode, the registers IFCR0 – IFCR15 are not present.

Note: In Master Mode, read access to IFMI register and read/write access to IFER and IFMR registers would result in ips_xfr_err being flagged.

Table 845. LINFlexD memory map

Address offset (hex)	Register description	Register classification(1)	Location
00	LIN Control register 1 (LINCR1)	Saf-Relv	*on page 1422*
04	LIN Interrupt Enable register (LINIER)	Saf-Relv	*on page 1424*
08	LIN Status register (LINSR)	No-Saf	*on page 1427*
0C	LIN Error Status register (LINESR)	No-Saf	*on page 1430*
10	UART Mode Control register (UARTCR)	Saf-Relv	*on page 1432*
14	UART Mode Status register (UARTSR)	No-Saf	*on page 1436*
18	LIN Timeout Control Status register (LINTCSR)	Saf-Relv	*on page 1438*
1C	LIN Output Compare register (LINOCR)	Saf-Relv	*on page 1439*
20	LIN Timeout Control register (LINTOCR)	Saf-Relv	*on page 1440*
24	LIN Fractional Baud Rate register (LINFBRR)	Saf-Relv	*on page 1441*
28	LIN Integer Baud Rate register (LINIBRR)	Saf-Relv	*on page 1442*

Table 845. LINFlexD memory map(Continued)

Address offset (hex)	Register description	Register classification(1)	Location
2C	LIN Checksum Field register (LINCFR)	Saf-Relv	on page 1443
30	LIN Control register 2 (LINCR2)	Saf-Relv	on page 1444
34	Buffer Identifier register (BIDR)	Saf-Relv	on page 1446
38	Buffer Data Register Least Significant (BDRL)	No-Saf	on page 1447
3C	Buffer Data Register Most Significant (BDRM)	No-Saf	on page 1448
40	Identifier Filter Enable register (IFER)	Saf-Relv	on page 1448
44	Identifier Filter Match index (IFMI)	Saf-Relv	on page 1449
48	Identifier Filter Mode register (IFMR)	Saf-Relv	on page 1449
4C(2)	Identifier Filter Control registers (IFCR0)	Saf-Relv	on page 1450
50(2)	Identifier Filter Control registers (IFCR1)	Saf-Relv	on page 1450
54(2)	Identifier Filter Control registers (IFCR2)	Saf-Relv	on page 1450
58(2)	Identifier Filter Control registers (IFCR3)	Saf-Relv	on page 1450
5C(2)	Identifier Filter Control registers (IFCR4)	Saf-Relv	on page 1450
60(2)	Identifier Filter Control registers (IFCR5)	Saf-Relv	on page 1450
64(2)	Identifier Filter Control registers (IFCR6)	Saf-Relv	on page 1450
68(2)	Identifier Filter Control registers (IFCR7)	Saf-Relv	on page 1450
6C(2)	Identifier Filter Control registers (IFCR8)	Saf-Relv	on page 1450
70(2)	Identifier Filter Control registers (IFCR9)	Saf-Relv	on page 1450
74(2)	Identifier Filter Control registers (IFCR10)	Saf-Relv	on page 1450
78(2)	Identifier Filter Control registers (IFCR11)	Saf-Relv	on page 1450
7C(2)	Identifier Filter Control registers (IFCR12)	Saf-Relv	on page 1450
80(2)	Identifier Filter Control registers (IFCR13)	Saf-Relv	on page 1450
84(2)	Identifier Filter Control registers (IFCR14)	Saf-Relv	on page 1450
88(2)	Identifier Filter Control registers (IFCR15)	Saf-Relv	on page 1450
8C(3)	Global Control register (GCR)	Saf-Relv	on page 1452
90(3)	UART Preset Timeout register (UARTPTO)	Saf-Relv	on page 1454
94(3)	UART Current Timeout register (UARTCTO)	No-Saf	on page 1455
98(3)	DMA Tx Enable register (DMATXE)	Saf-Relv	on page 1456
9C(3)	DMA Rx Enable register (DMARXE)	Saf-Relv	on page 1457

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.

50.4.2 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.

50.4.2.1 LIN Control Register 1 (LINCR1)

LINCR1 consists of control bits used to configure features of the LINFlexD.

Address: 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	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CCD(1)	CFD(1)	LASE(1)	AUTOWU(1)	MBL(1)				BF(1)	0	LBKM(1)	MME(1)	SSBL(1)	RBLM(1)	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0/1(2)	0	0	1	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. When slave = 0, this field reads '1' and cannot be programmed.

Figure 857. LIN Control Register 1 (LINCR1)

Table 846. LINCR1 field descriptions

Field	Description
0–15	Reserved Read returns 0
16 CCD	Checksum Calculation disable 0 Checksum calculation is done by hardware When this bit is reset the LINCFR register is read-only. 1 Checksum calculation is disabled When this bit is set the LINCFR register is read/write. User can program this register to send a software calculated checksum/CRC (provided CFD is reset). This bit can be written during Initialization mode only. It is read-only in Normal mode.
17 CFD	Checksum field disable 0 Checksum field is sent after the required number of data bytes are sent 1 No checksum field is sent in the frame This bit can be configured during Initialization mode only. It is read-only in Normal mode.
18 LASE	LIN Auto Synchronization Enable 0 Auto synchronization disabled 1 Auto synchronization enabled This bit can be programmed in Initialization mode only. It is read-only in Normal mode. **Note:** If generic auto_sync = 0, then this bit will always read a 0.

Table 846. LINCR1 field descriptions(Continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>19
AUTOWU</td><td>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.</td></tr>
<tr><td>20–23
MBL[3:0]</td><td>Master Break Length
These bits choose the length of the Sync break to be generated by the master.

<table>
<tr><th>MBL[3:0]</th><th>Break length</th></tr>
<tr><td>0</td><td>10-bit</td></tr>
<tr><td>1</td><td>11-bit</td></tr>
<tr><td>2</td><td>12-bit</td></tr>
<tr><td>3</td><td>13-bit</td></tr>
<tr><td>4</td><td>14-bit</td></tr>
<tr><td>5</td><td>15-bit</td></tr>
<tr><td>6</td><td>16-bit</td></tr>
<tr><td>7</td><td>17-bit</td></tr>
<tr><td>8</td><td>18-bit</td></tr>
<tr><td>9</td><td>19-bit</td></tr>
<tr><td>A</td><td>20-bit</td></tr>
<tr><td>B</td><td>21-bit</td></tr>
<tr><td>C</td><td>22-bit</td></tr>
<tr><td>D</td><td>23-bit</td></tr>
<tr><td>E</td><td>36-bit (cooling)</td></tr>
<tr><td>F</td><td>50-bit</td></tr>
</table>

These bits can be programmed in Initialization mode only. They are read-only in Normal mode.</td></tr>
<tr><td>24
BF</td><td>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.</td></tr>
<tr><td>25</td><td>Reserved</td></tr>
</table>

Table 846. LINCR1 field descriptions(Continued)

Field	Description
26 LBKM	Loop Back mode 0 Loop Back Mode disabled 1 Loop Back mode enabled Refer to *Section 50.3.2.2, Test mode*. **Note:** This bit can be programmed only in Initialization mode. This bit is utilized in UART mode also.
27 MME	Master mode enable 0 Slave Mode 1 Master Mode This bit can be programmed in Initialization mode only. It is read-only in Normal mode. **Note:** 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.

50.4.2.2 LIN Interrupt enable register (LINIER)

Address : 0004h 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									
W	SZIE	OCIE	BEIE	CEIE	HEIE			FEIE	BOIE	LSIE	WUIE	DBFIE	DBEIE_TOIE	DRIE	DTIE	HRIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 858. LIN Interrupt enable register (LINIER)

Table 847. LINIER field descriptions

Field	Description
0–15	Reserved Read returns 0.
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.
21–22	Reserved Read returns 0.
23 FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generated if Frame Error Flag (FEF) bit is set in LINESR or UARTSR

Table 847. LINIER field descriptions(Continued)

Field	Description
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.
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.

50.4.2.3 LIN Status Register (LINSR)

This register consists of status bits indicating the state of the LINFlexD hardware.

Address: 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	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 859. LIN Status Register (LINSR)

Table 848. LINSR field descriptions

Field	Description
0–11	Reserved Read returns 0
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 848. LINSR field descriptions(Continued)

Field	Description
16–19 LINS[3:0]	LIN state 0: Sleep mode LINFlexD is in Sleep mode, to save power consumption. 1: Init, 2: 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. 3: 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. **Note:** 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. 4: 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. 5: 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. 6: Identifier Field In Slave mode, a valid sync field has been received. Receiving ID field. In Master mode, identifier transmission is ongoing. 7: 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. 8: Data Reception/Data Transmission In Receiver mode, reception ongoing. In Transmitter mode, response transmission ongoing. 9: *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.
20–21	Reserved Read returns 0.
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 848. LINSR field descriptions(Continued)

Field	Description
24 RXBUSY	Receiver Busy flag 0 Receiver is idle 1 Reception ongoing In Slave mode after header reception, if DIR bit is reset and reception starts, then this bit is set. In this case user cannot set the DTRQ bit.
25 RDI	LIN Receive signal This bit reflects the current status of the Rx pin. **Note:** After reset is released, RDI reflects the actual value of Rx pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the Rx pin and when: – The slave is in Sleep mode – The master is in Sleep mode or Idle mode It can be cleared by software. It gets reset by hardware in Initialization mode.
27 DBFF	Data Buffer full flag This bit is set by hardware when the buffer is full (8 bytes received) and DFL > 7. It should be cleared by software. This bit is also reset by hardware in Initialization mode.
28 DBEF	Data Buffer empty flag This bit is set by hardware when the buffer is empty (8 bytes have been transmitted) and DFL > 7. It should be cleared by software after data buffer is refilled by software. Transmission of next 8 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.

50.4.2.4 LIN Error Status Register (LINESR)

Address: 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	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 860. LIN error status register (LINESR)

Refer to *Section 50.3.2.5, Errors* for detailed description of all errors.

Table 849. LINESR field descriptions

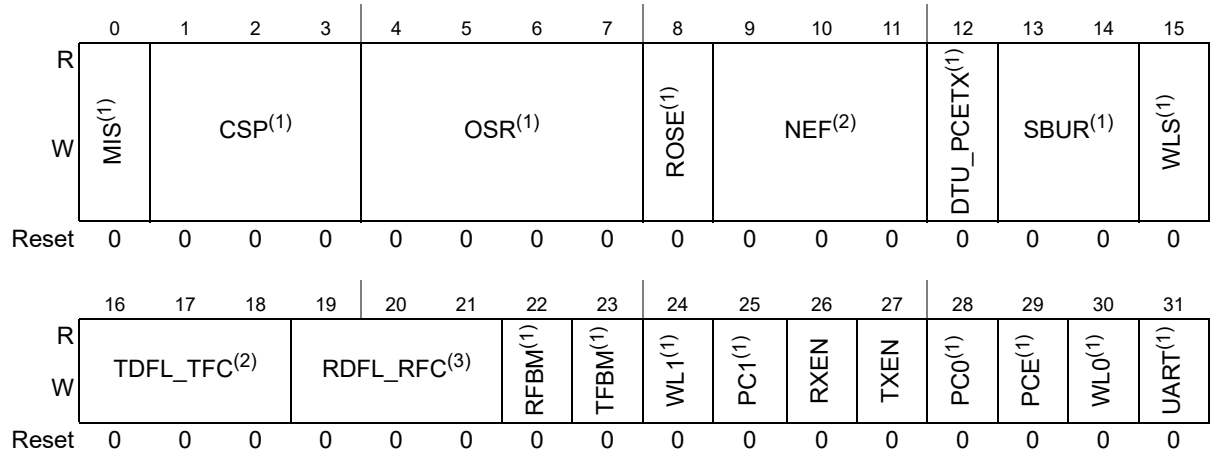
Field	Description
0–15	Reserved Read returns 0
16 SZF	Stuck at Zero flag This bit is set when there is a stuck-at-zero timeout error. It should be reset by software.
17 OCF	Output Compare Flag 0 No output compare event occurred 1 The content of the counter has matched the content of OCR1 or OCR2 This 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. If generic *slave* = 0, then this bit will always read a 0, and cannot be programmed.

Table 849. 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.
25–30	Reserved Read returns 0.
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.

50.4.2.5 UART Mode Control Register (UARTCR)

Address: 0010h Access: User read/write



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 861. UART Mode Control Register (UARTCR)

Table 850. 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** / **CSP** 4 / 2, 3 5 / 2, 3, 4 6 / 3, 4, 5 8 / 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.

Table 850. UARTCR field descriptions(Continued)

Field	Description
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.
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.

Table 850. UARTCR field descriptions(Continued)

Field	Description
16–18 TDFL_TFC	Transmitter Data Field Length/TX FIFO Counter TDFL defines the number of bytes to be transmitted in UART buffer mode (TFBM = 0). TDFL is a read/write field. Bit 15 is reserved and not implemented. When the UART data length is configured as a halfword (WL = 10 or WL = 11), only the configurations TDFL = x01 or TDFL = x11 are allowed. x00 1 byte x01 2 bytes x10 3 bytes x11 4 bytes TFC contains the number of entries (bytes) of the Tx FIFO in UART FIFO mode (TFBM = 1). 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 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 defines the number of bytes to be received in UART buffer mode (RFBM = 0). RDFL 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 = x01 or RDFL = x11 are allowed. x00 1 byte x01 2 bytes x10 3 bytes x11 4 bytes RFC contains the number of entries (bytes) of the Rx FIFO in UART FIFO mode (RFBM = 1). 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 should be programmed to be greater than or equal to NEF (number of expected frames). In FIFO mode, there is no such constraint.
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.

Table 850. UARTCR field descriptions(Continued)

Field	Description
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 850. 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 = 11015 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.

50.4.2.6 UART Mode Status Register (UARTSR)

Address: 0014h 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 862. UART Mode Status Register (UARTSR)

Table 851. UARTSR field descriptions

Field	Description
0–15	Reserved Read returns 0
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.

Table 851. UARTSR field descriptions(Continued)

Field	Description
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. **Note:** 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.
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.

Table 851. UARTSR field descriptions(Continued)

Field	Description
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 is set by hardware in UART buffer mode (RFBM = 0) and indicates that the number of bytes programmed in RDFL have been received. DRF should be cleared by software. An interrupt will be generated if the DRIE bit in LINIER is set. DRF 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 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. RFE is set by hardware in UART FIFO mode (RFBM = 1) when the RX FIFO is empty. 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 is set by hardware in UART buffer mode (TFBM = 0) and indicates that data transmission is completed. DTF 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. TFF is set by hardware in UART FIFO mode (TFBM = 1) when TX FIFO is full. 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.

50.4.2.7 LIN Time-Out Control Status Register (LINTCSR)

This register contains control and status bits for the timeout feature.

Address: 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	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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 863. LIN Time-Out Control Status Register (LINTCSR)

Table 852. LINTCSR field descriptions

Field	Description
0–20	Reserved Read returns 0.
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[0:7]	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 $\geq$ 5.

50.4.2.8 LIN Output Compare Register (LINOCR)

This register contains the value to be compared to the LINTOCR value. This register is writable by software only in Output Compare Mode.

Address: 001Ch 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 864. LIN Output Compare Register (LINOCR)

Table 853. LINOCR field descriptions

Field	Description
0–15	Reserved Read returns 0.
16–23 OC2[7:0]	Output compare value 2
24–31 OC1[7:0]	Output compare value 1

50.4.2.9 LIN Time-Out Control Register (LINTOCR)

Address: 0020h 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 865. LIN Time-Out Control Register (LINTOCR)

Table 854. LINTOCR field descriptions

Field	Description
0–19	Reserved Read returns 0.
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 $T_{Response_Maximum} = 1.4 \times T_{Response_Nominal}$
24	Reserved Read returns 0.
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.

50.4.2.10 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: 0024h 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 866. LIN Fractional Baud Rate Register (LINFBRR)

Table 855. LINFBRR field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–27</td><td>Reserved
Read returns 0</td></tr>
<tr><td>28–31
FBR[3:0]</td><td>Fractional Baud rates
<table>
<tr><th>LINFBRR[4:0]</th><th>Fraction (LDIV)</th></tr>
<tr><td>0h</td><td>0</td></tr>
<tr><td>1h</td><td>1/16</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>Eh</td><td>14/16</td></tr>
<tr><td>Fh</td><td>15/16</td></tr>
</table></td></tr>
</table>

50.4.2.11 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: 0028h Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	IBR(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IBR(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 867. LIN Integer Baud Rate Register (LINIBRR)

Table 856. LINIBRR field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–11</td><td>Reserved
Read returns 0.</td></tr>
<tr><td>12–31
IBR[19:0]</td><td>Integer Baud rates
These bits along with the fractional baud rate bits decide the LIN baud rate.

<table>
<tr><th>LINIBRR[19:0]</th><th>Mantissa (LDIV)</th></tr>
<tr><td>00000h</td><td>LIN clock disabled</td></tr>
<tr><td>00001h</td><td>1</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>FFFFEh</td><td>1048574</td></tr>
<tr><td>FFFFFh</td><td>1048575</td></tr>
</table>
</td></tr>
</table>

50.4.2.12 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: 002Ch 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 868. LIN Checksum Field Register (LINCFR)

Table 857. LINCFR field descriptions

Field	Description
0–23	Reserved Read returns 0.
24–31 CF[7:0]	Checksum bits When the CCD bit is reset these bits are read-only and are calculated by hardware. When the CCD bit is set, these bits can be written by software.

CFD	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

50.4.2.13 LIN Control Register 2 (LINCR2)

This register includes control status bits related to buffer operations.

Address: 0030h 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 869. LIN Control Register 2 (LINCR2)

Table 858. LINCR2 field descriptions

Field	Description
0–15	Reserved Read returns 0.
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. 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.

Table 858. LINCR2 field descriptions(Continued)

Field	Description
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.
24–31	Reserved Read returns 0.

50.4.2.14 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: 0034h 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 870. Buffer Identifier Register (BIDR)

Table 859. BIDR field descriptions

Field	Description
0–15	Reserved Read returns 0.
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.

Table 859. BIDR field descriptions(Continued)

Field	Description
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives the data and copy them in the BDR registers 1 LINFlexD transmits the 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.
24–25	Reserved Forced to 0 by hardware.
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’).

50.4.2.15 Buffer Data Register Least Significant (BDRL)

This register is a part of an 8-byte data buffer.

Address: 0038h 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 871. Buffer Data Register Least Significant (BDRL)

Table 860. 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

50.4.2.16 Buffer Data Register Most Significant (BDRM)

This register is a part of an 8-byte data buffer.

Address: 003Ch 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 872. Buffer Data Register Most Significant (BDRM)

Table 861. 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 5of the data field
24–31 DATA4[7:0]	Data Byte 4 Data byte 4 of the data field

50.4.2.17 Identifier Filter Enable Register (IFER)

This register enables/disables a particular filter.

Address: 0040h 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 873. Identifier Filter Enable Register (IFER)

Table 862. IFER field descriptions

Field	Description
0–15	Reserved Read returns 0
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 (2n+1) have no effect on the corresponding filters as they act as mask for the Identifier 2n. 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.

50.4.2.18 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: 0044h 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 874. Identifier Filter Match Index (IFMI)

Table 863. IFMI field descriptions

Field	Description
0–26	Reserved Read returns 0.
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 50.3.2.4, Slave mode* for more details).

50.4.2.19 Identifier Filter Mode Register (IFMR)

This register configures the modes of filters.

Address: 0048h 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 875. Identifier Filter Mode Register (IFMR)

Table 864. IFMR field descriptions

Field	Description
0–23	Reserved Read returns 0.
24–31 IFM[7:0]	Filter mode 0 Filters 2n and 2n+1 are in identifier list mode 1 Filters 2n and 2n+1 are in mask mode, where filter 2n+1 is the mask for filter 2n

50.4.2.20 Identifier Filter Control Register (IFCR2n)

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: (4C+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 876. Identifier Filter Control Register (IFCR2n)

Table 865. IFCR2n field descriptions

Field	Description
0–15	Reserved Read returns a 0.
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.
24–25	Reserved Forced to 0 by hardware.
26–31 ID[5:0]	Identifier Identifier part of the identifier field without the identifier parity.

50.4.2.21 Identifier Filter Control Register (IFCR2n+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 IFCR2n in Identifier mask mode.

Address: (50+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 877. Identifier Filter Control Register (IFCR2n+1)

Table 866. IFCR2n+1 field descriptions

Field	Description
0–15	Reserved Read returns a 0
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.
24–25	Reserved Forced to 0 by hardware.
26–31 ID[5:0]	Identifier Identifier part of the identifier field without the identifier parity.

50.4.2.22 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: 4C+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						
W											TDFBM(1)	RDFBM(1)	TDLIS(1)	RDLIS(1)	STOP(1)	SR(2)
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 can be written only by software in *Initialization* mode. This field reads 0 by software.
3. Refer to device configuration chapter to see the number of filters and value of slave used in the device.

Figure 878. Global Control Register (GCR)

Table 867. GCR field descriptions

Field	Description
0–25	Reserved Forced to 0 by hardware
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB in both UART and LIN modes. 0 The first bit of transmitted data is LSB — in other words, the first bit transmitted is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of transmitted data is MSB — in other words, the first bit transmitted is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB both in UART and LIN modes. 0 The first bit of received data is LSB — in other words, the first bit received is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of received data is MSB — in other words, 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 867. GCR field descriptions(Continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>30
STOP</td><td>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</td></tr>
<tr><td>31
SR</td><td>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 table describes, the register fields reset by SR:
<table>
<tr><th>Register</th><th>Comment</th></tr>
<tr><td>LINSR</td><td>All fields except RXbusy and AUTOSYNC_COMP are reset</td></tr>
<tr><td>LINESR</td><td>All fields are reset</td></tr>
<tr><td>LINTCSR</td><td>Only CNT[0:7] is reset</td></tr>
<tr><td>UARTSR</td><td>All fields except RFE & TFF are reset</td></tr>
<tr><td>UARTCR</td><td>Only TFC & RFC are reset</td></tr>
<tr><td>UARTCTO</td><td>All fields are reset</td></tr>
</table>
</td></tr>
</table>

50.4.2.23 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 879. UART Preset Timeout Register (UARTPTO)

Table 868. UARTPTO field descriptions

Field	Description
0–19	Reserved Read returns 0.
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.

50.4.2.24 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 50.4.2.23, UART Preset Timeout Register (UARTPTO)* 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 880. UART Current Timeout Register (UARTCTO)

Table 869. UARTCTO field descriptions

Field	Description
0–19	Reserved Read returns 0.
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.

50.4.2.25 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 881. DMA Tx Enable Register (DMATXE)

Table 870. DMATXE field descriptions

Field	Description
0 – (2**TX_CH_NUM–1)	Reserved Read returns 0. **Note:** Refer to the device configuration chapter for the value of TX_CH_NUM used in this device.
(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. **Note:** Refer to the device configuration chapter for the value of TX_CH_NUM used in this device.

1. ** stands for exponentiation.

50.4.2.26 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 W	DRE															
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 882. DMA Rx Enable Register (DMARXE)

Table 871. DMARXE field descriptions

Field	Description
0 – (2**RX_CH_NUM–1)	Reserved Read returns 0. **Note:** Refer to the device configuration chapter for the value of RX_CH_NUM used in this device.
(2**RX_CH_NUM) – 31 DRE[2**RX_CH_NUM–1:0](1)	DMA Rx channel Y enable 0 DMA Rx channel Y disabled 1 DMA Rx channel Y enabled **Note:** The actual size of the register DMARXE depends on the value of the static parameter RX_CH_NUM. When DMARXE = 0x00000000, the DMA RX interface FSM is forced (soft reset) in Idle state. **Note:** Refer to the device configuration chapter for the value of RX_CH_NUM used in this device.

1. ** stands for exponentiation.

50.4.3 Register map

50.4.3.1 IPS bus

Table 872. LINFlexD register map

Addr offset	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
0	LINCR1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		CCD	CFD	LASE	AUTOWU	MBL3	MBL2	MBL1	MBL0	BF	SLFM	LBKM	MME	SSBL	RBLM	SLEEP	INIT
4	LINIER	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		SZIE	OCIE	BEIE	CEIE	HEIE	–	–	FEIE	BOIE	LSIE	WUIE	DBFIE	DBEIE/TOIE	DRIE	DTIE	HRIE
8	LINSR	—	—	—	—	—	—	—	—	—	—	—	—	AUTOSYNC_COMP	RDC		
		LINS3	LINS2	LINS1	LINS0	—	—	RMB	DRBNE	RXbusy	RDI	WUF	DBFF	DBEF	DRF	DTF	HRF
C	LINESR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		SZF	OCF	BEF	CEF	SFEF	SDEF	IDPEF	FEF	BOF	—	—	—	—	—	—	NF
10	UARTCR	MIS	CSP			OSR				ROSE	NEF			DTU	SBUR		WLS
		TDFL/TFC			RDFL/RFC			RFBM	TFBM	WL1	PC1	RxEn	TxEn	PC0	PCE	WL0	UART
14	UARTSR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		SZF	OCF	PE3	PE2	PE1	PE0	RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRF/RFE	DTF/TFF	NF
18	LINTCSR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	MODE	IOT	TOCE	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
1C	LINOCR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		OC27	OC26	OC25	OC24	OC23	OC22	OC21	OC20	OC17	OC16	OC15	OC14	OC13	OC12	OC11	OC10
20	LINTOCR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	RTO3	RTO2	RTO1	RTO0	–	HTO6	HTO5	HTO4	HTO3	HTO2	HTO1	HTO0

Table 872. LINFlexD register map(Continued)

Addr offset	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
24	LINFBRR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	—	—	—	—	—	—	—	FBR3	FBR2	FBR1	FBR0
28	LINIBRR	—	—	—	—	—	—	—	—	—	—	—	—	IBR 19	IBR 18	IBR 17	IBR 16
		IBR 15	IBR 14	IBR 13	IBR 12	IBR 11	IBR 10	IBR9	IBR8	IBR7	IBR6	IBR5	IBR4	IBR3	IBR2	IBR1	IBR0
2C	LINCFR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	—	—	—	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
30	LINCR2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		TBDE	IOBE	IOPE	WURQ	DDRQ	DTRQ	ABRQ	HTRQ	—	—	—	—	—	—	—	—
34	BIDR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		DFL5	DFL4	DFL3	DFL2	DFL1	DFL0	DIR	CCS	—	—	ID5	ID4	ID3	ID2	ID1	ID0
38	BDRL	DATA3	DATA3	DATA3	DATA3	DATA3	DATA3	DATA3	DATA3	DATA2	DATA2	DATA2	DATA2	DATA2	DATA2	DATA2	DATA2
		DATA1	DATA1	DATA1	DATA1	DATA1	DATA1	DATA1	DATA1	DATA0	DATA0	DATA0	DATA0	DATA0	DATA0	DATA0	DATA0
3C	BDRM	DATA7	DATA7	DATA7	DATA7	DATA7	DATA7	DATA7	DATA7	DATA6	DATA6	DATA6	DATA6	DATA6	DATA6	DATA6	DATA6
		DATA5	DATA5	DATA5	DATA5	DATA5	DATA5	DATA5	DATA5	DATA4	DATA4	DATA4	DATA4	DATA4	DATA4	DATA4	DATA4
40	IFER	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		FACT 15	FACT 14	FACT 13	FACT 12	FACT 11	FACT 10	FACT 9	FACT 8	FACT 7	FACT 6	FACT 5	FACT 4	FACT 3	FACT 2	FACT 1	FACT 0
44	IFMI	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	—	—	—	—	—	—	IFMI4	IFMI3	IFMI2	IFMI1	IFMI0
48	IFMR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	—	—	—	IFM7	IFM6	IFM5	IFM4	IFM3	IFM2	IFM1	IFM0
4C+ 8×n	IFCR2n	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		DFL5	DFL4	DFL3	DFL2	DFL1	DFL0	DIR	CCS	—	—	ID5	ID4	ID3	ID2	ID1	ID0

Table 872. LINFlexD register map(Continued)

Addr offset	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
50+8×n	IFCR2n+1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		DFL5	DFL4	DFL3	DFL2	DFL1	DFL0	DIR	CCS	—	—	ID5	ID4	ID3	ID2	ID1	ID0
4C+8×n+04h	GCR	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	—	—	—	—	—	—	TDFBM	RDFBM	TDLIS	RDLIS	STOP	SR
GCR offset +04h	UART PTO	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	PTO11	PTO10	PTO9	PTO8	PTO7	PTO6	PTO5	PTO4	PTO3	PTO2	PTO1	PTO0
GCR offset +08h	UART CTO	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		—	—	—	—	CTO11	CTO10	CTO9	CTO8	CTO7	CTO6	CTO5	CTO4	CTO3	CTO2	CTO1	CTO0
GCR offset +0Ch	DMATXE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		DTE15	DTE14	DTE13	DTE12	DTE11	DTE10	DTE9	DTE8	DTE7	DTE6	DTE5	DTE4	DTE3	DTE2	DTE1	DTE0
GCR offset +10h	DMARXE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		DRE15	DRE14	DRE13	DRE12	DRE11	DRE10	DRE9	DRE8	DRE7	DRE6	DRE5	DRE4	DRE3	DRE2	DRE1	DRE0

50.5 Programming considerations

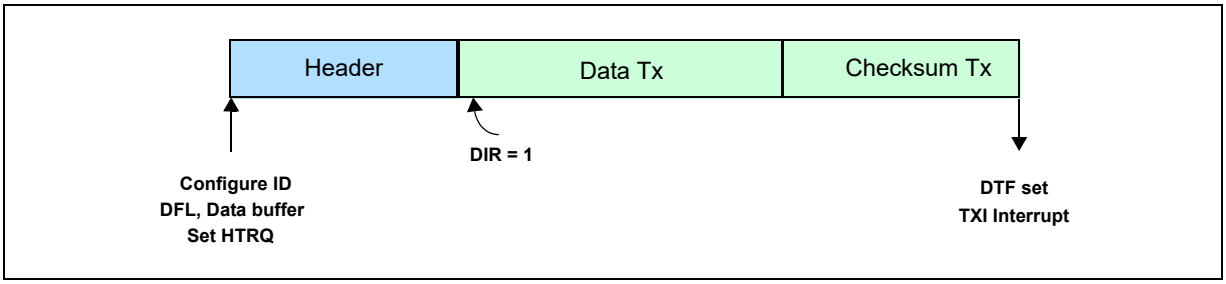
The next subsections describe the various configurations in which the LINFlexD module can be used.

50.5.1 Master node

LINFlexD acts as Master when the MME bit is set.

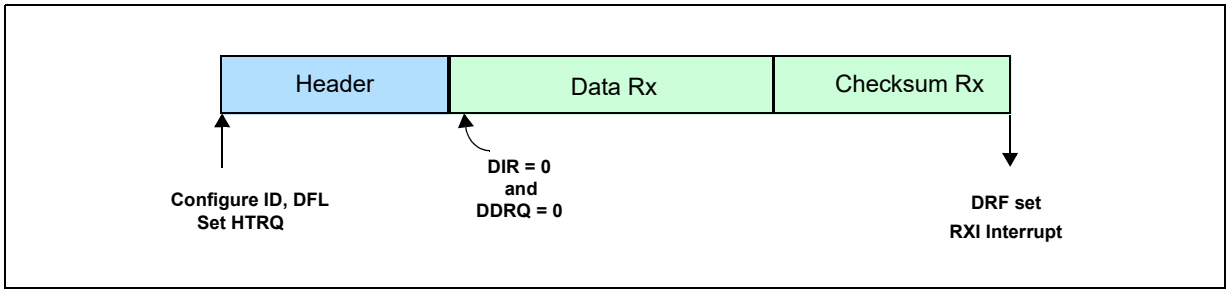
50.5.1.1 Transmitter

Figure 883. Master node — transmitter configuration



50.5.1.2 Receiver

Figure 884. Master node — receiver configuration



50.5.1.3 Transmitter, Bit Error

Figure 885. Master node — transmitter, bit error configuration

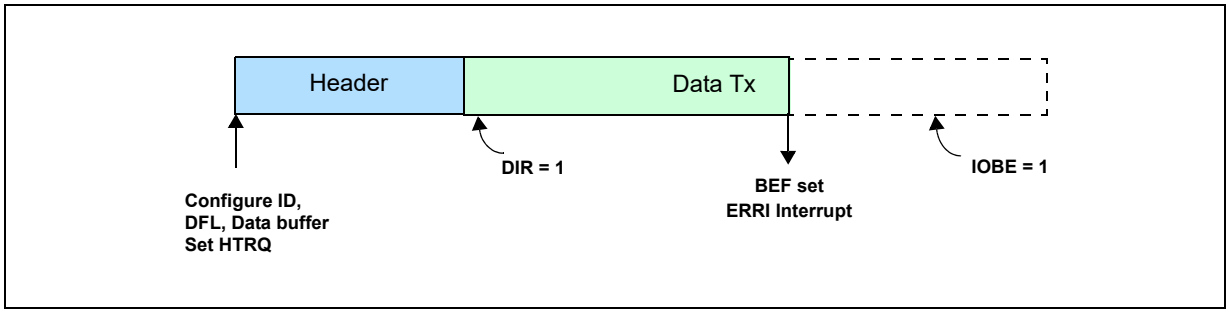
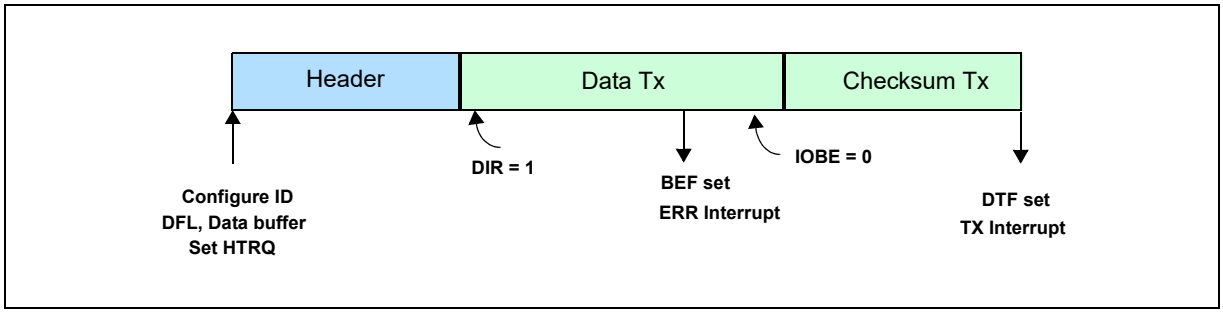
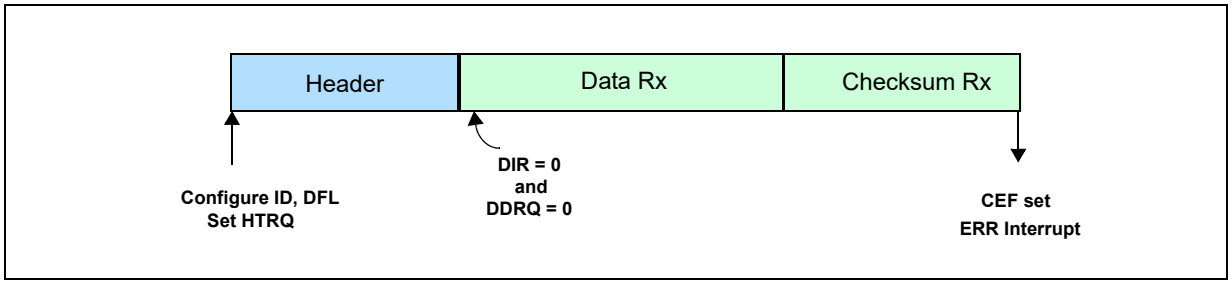


Figure 886. Master node — transmitter, checksum error configuration



50.5.1.4 Receiver, Checksum Error

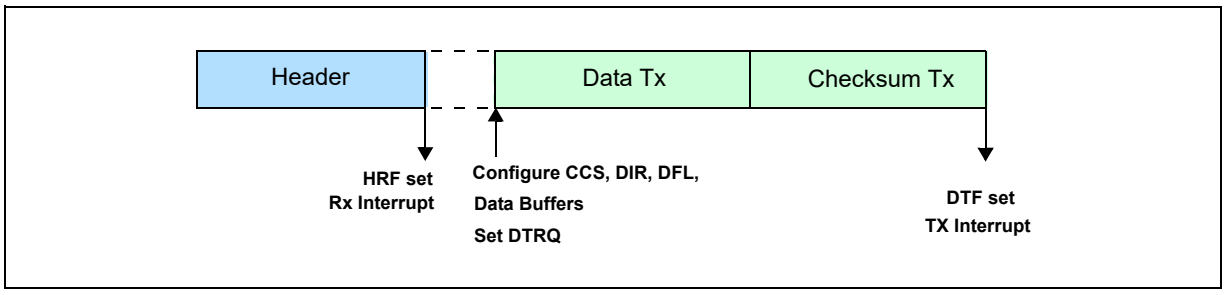
Figure 887. Master node — receiver, checksum error configuration



50.5.2 Slave node

50.5.2.1 Transmitter (no identifier filters)

Figure 888. Slave node — transmitter (no identifier filters) configuration



50.5.2.2 Receiver (no identifier filters)

Figure 889. Slave node — receiver (no identifier filters) configuration

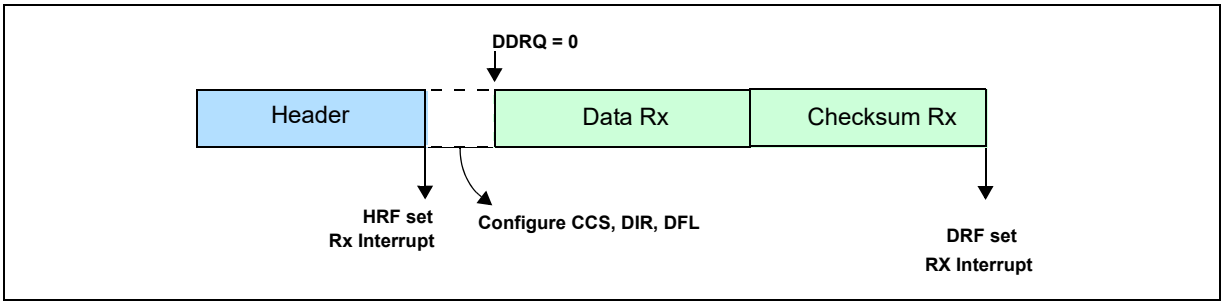
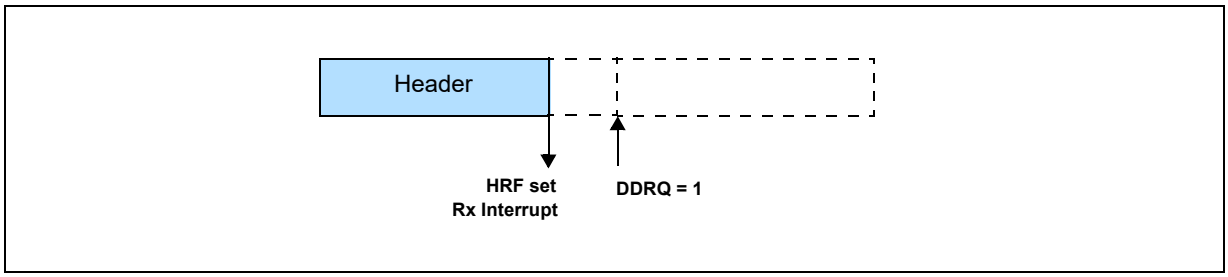
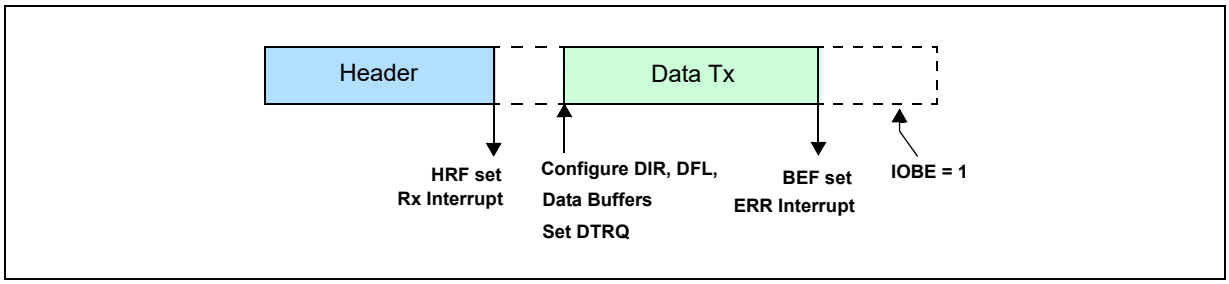


Figure 890. Slave node — receiver (no identifier filters) configuration



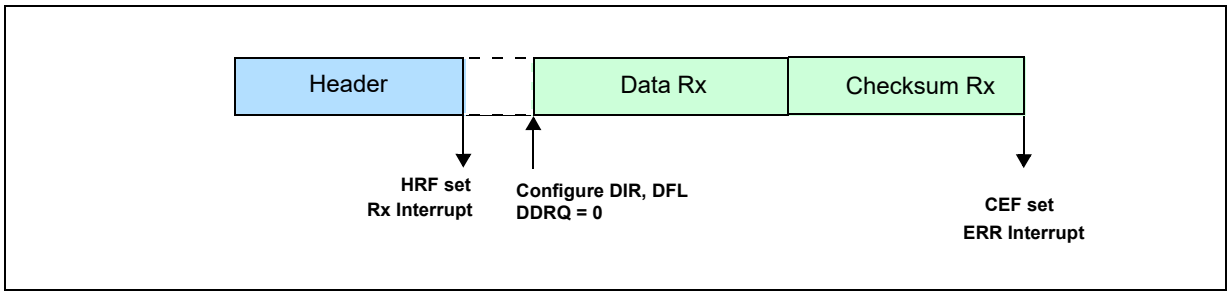
50.5.2.3 No filters, transmitter, bit error

Figure 891. Slave node — No filters, transmitter, bit error configuration



50.5.2.4 No filters, receiver, checksum error

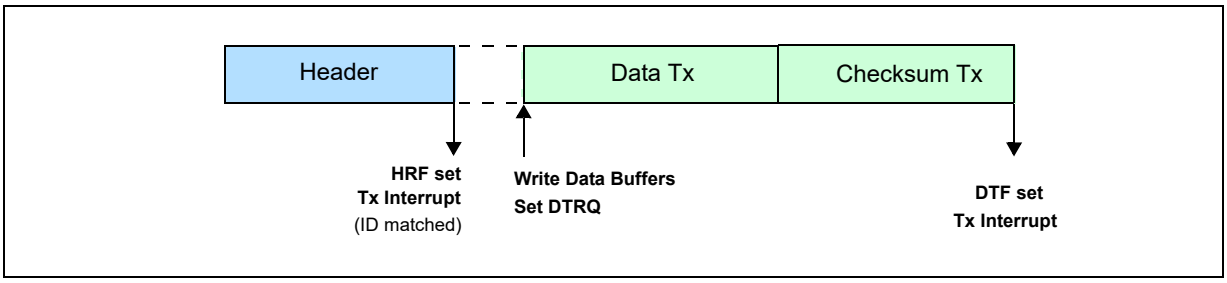
Figure 892. Slave node — No filters, receiver, checksum error configuration



50.5.2.5 At least one TX filter, BF is reset, ID matches filter

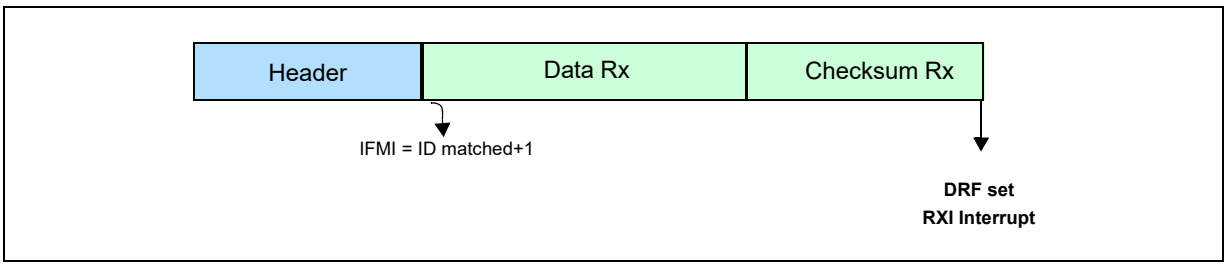
This configuration can be used in case slave never receives data, for example, sensor.

Figure 893. Slave node — one TX filter configuration



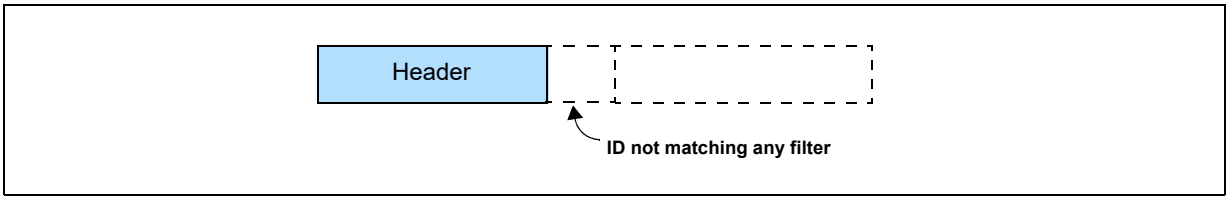
50.5.2.6 At least one RX filter, BF reset, ID matches filter

Figure 894. Slave node — one RX filter configuration



50.5.2.7 RX only, TX only, RX & TX filters, ID not matching filter, BF reset

Figure 895. RX only, TX only, RX & TX filters configuration



50.5.2.8 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 896. TX filter, BF is set configuration — ID has matched

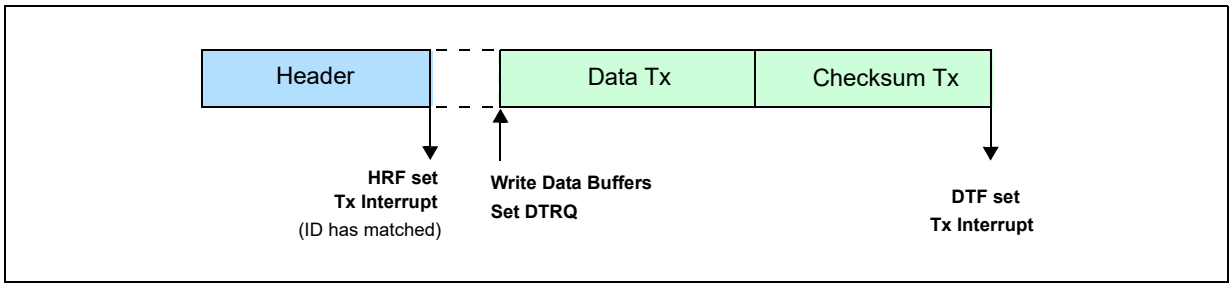
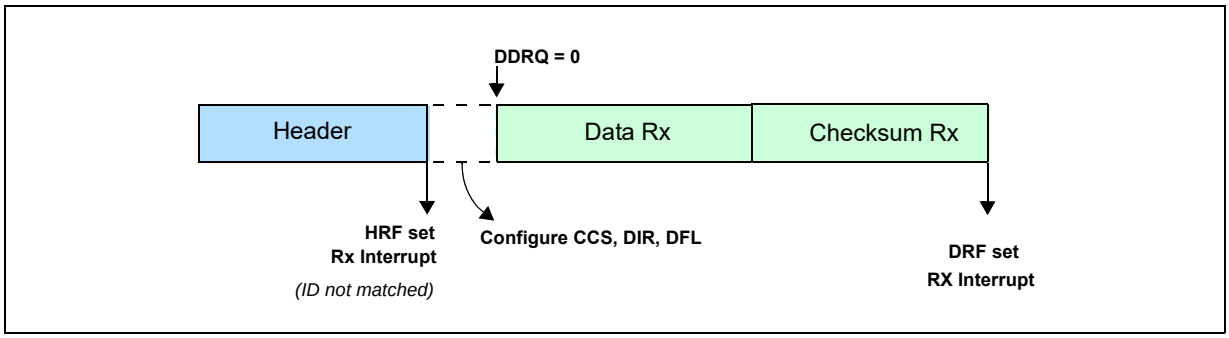


Figure 897. TX filter, BF is set configuration — ID not matched



50.5.2.9 RX filter, BF is set

Figure 898. RX filter, BF is set configuration — ID matched

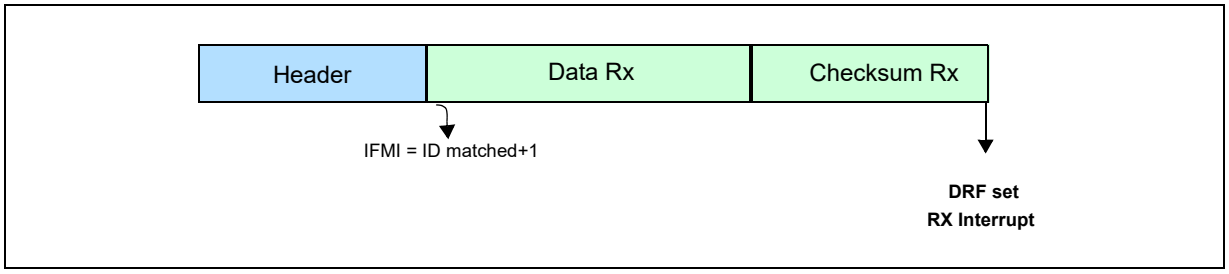


Figure 899. RX filter, BF is set configuration — ID not matched (ID is Rx)

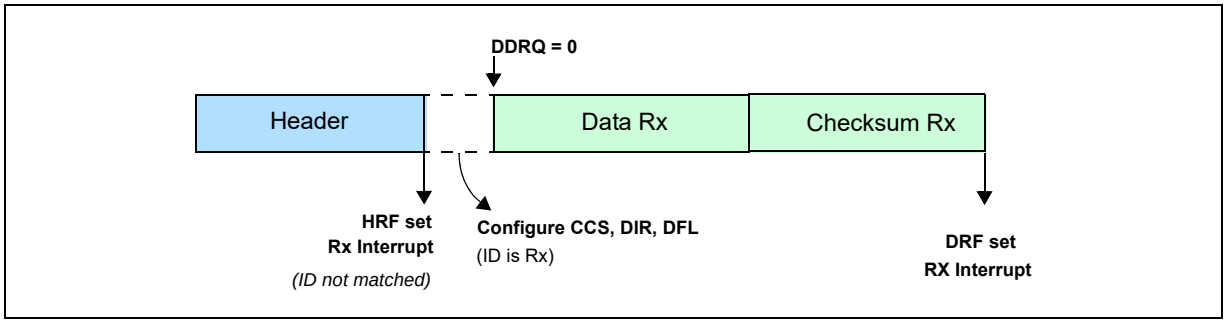
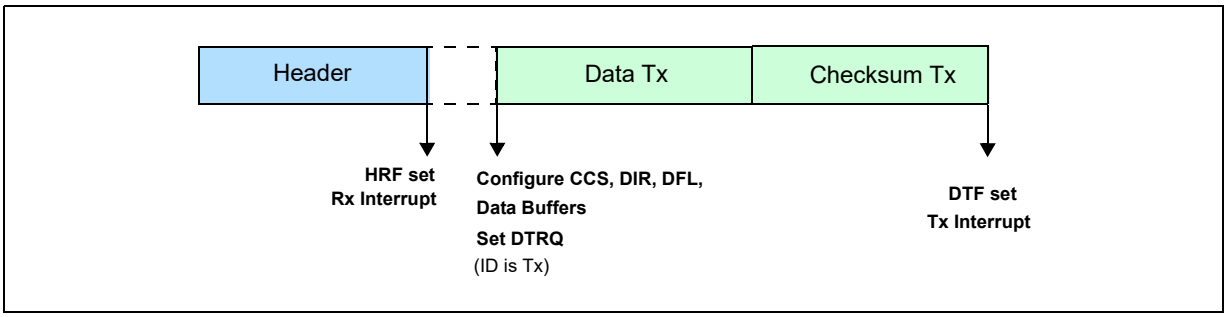


Figure 900. RX filter, BF is set configuration — (ID is Tx)



50.5.2.10 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 901. TX filter, RX filter, BF set configuration — (ID matched)

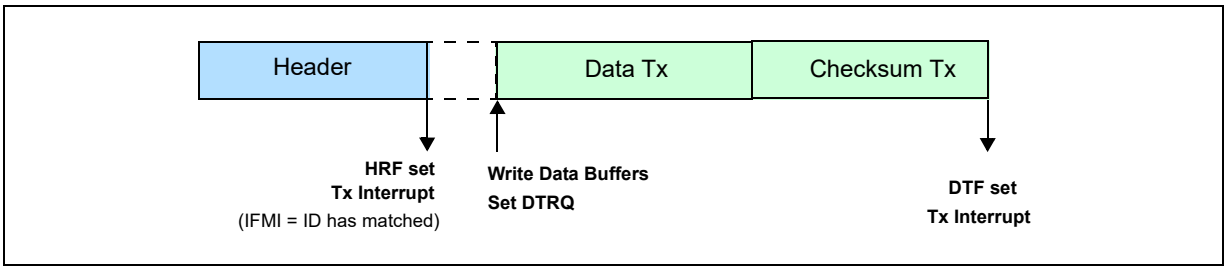


Figure 902. TX filter, RX filter, BF set configuration — (ID matched + 1)

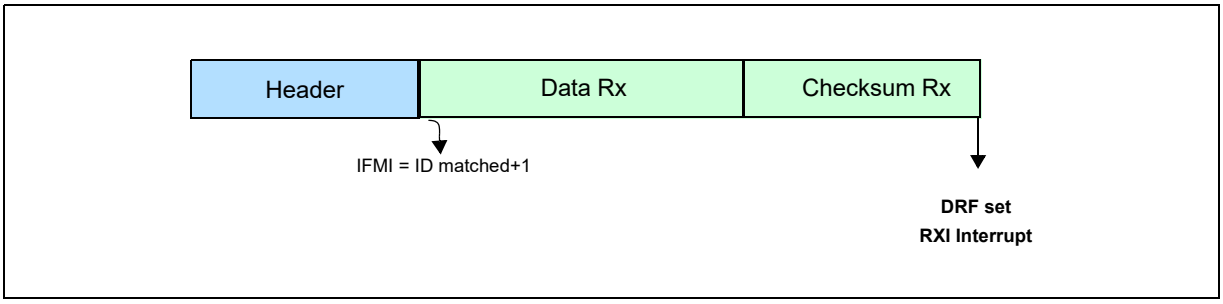
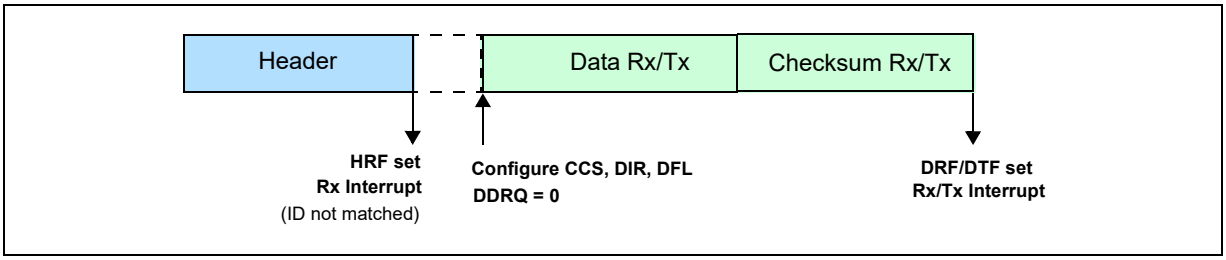


Figure 903. TX filter, RX filter, BF set configuration—(ID not matched, RX/TX Interrupt)



50.5.3 Extended frames

Figure 904. Extended frames (TX Interrupt)

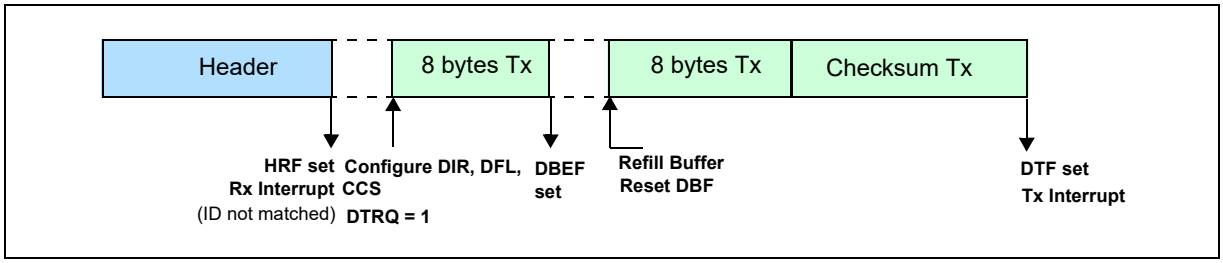
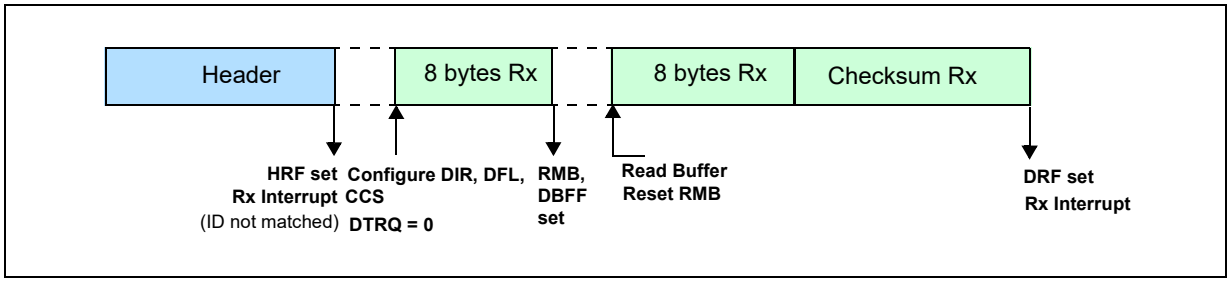


Figure 905. Extended frames (RX Interrupt)



50.5.4 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 906. Response timeout

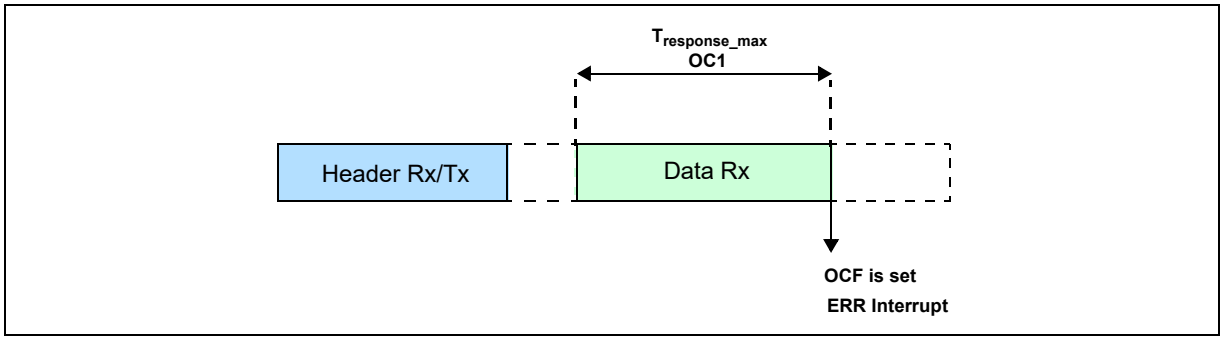


Figure 907. Frame timeout

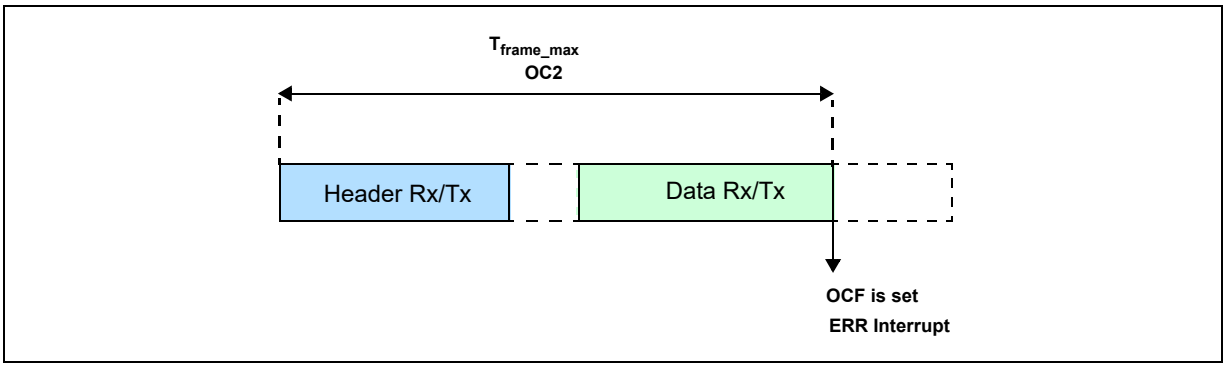
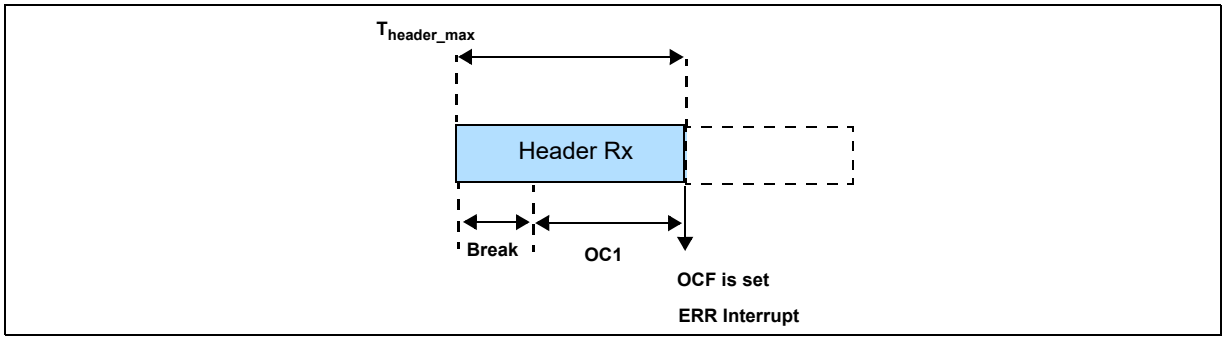
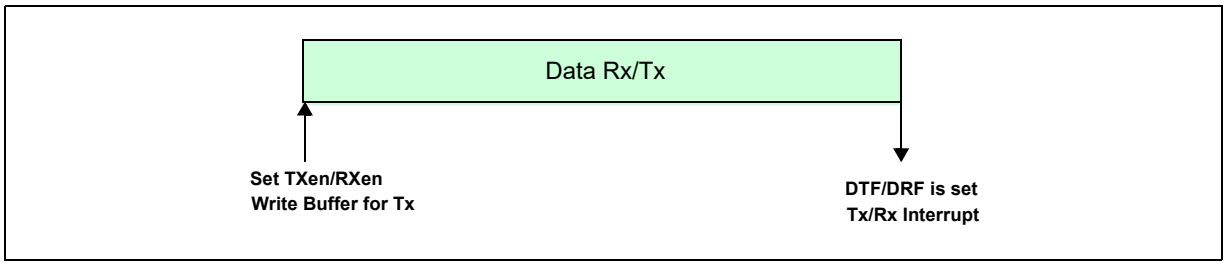


Figure 908. Header timeout



50.5.5 UART mode

Figure 909. UART mode configuration



50.5.6 Interrupts

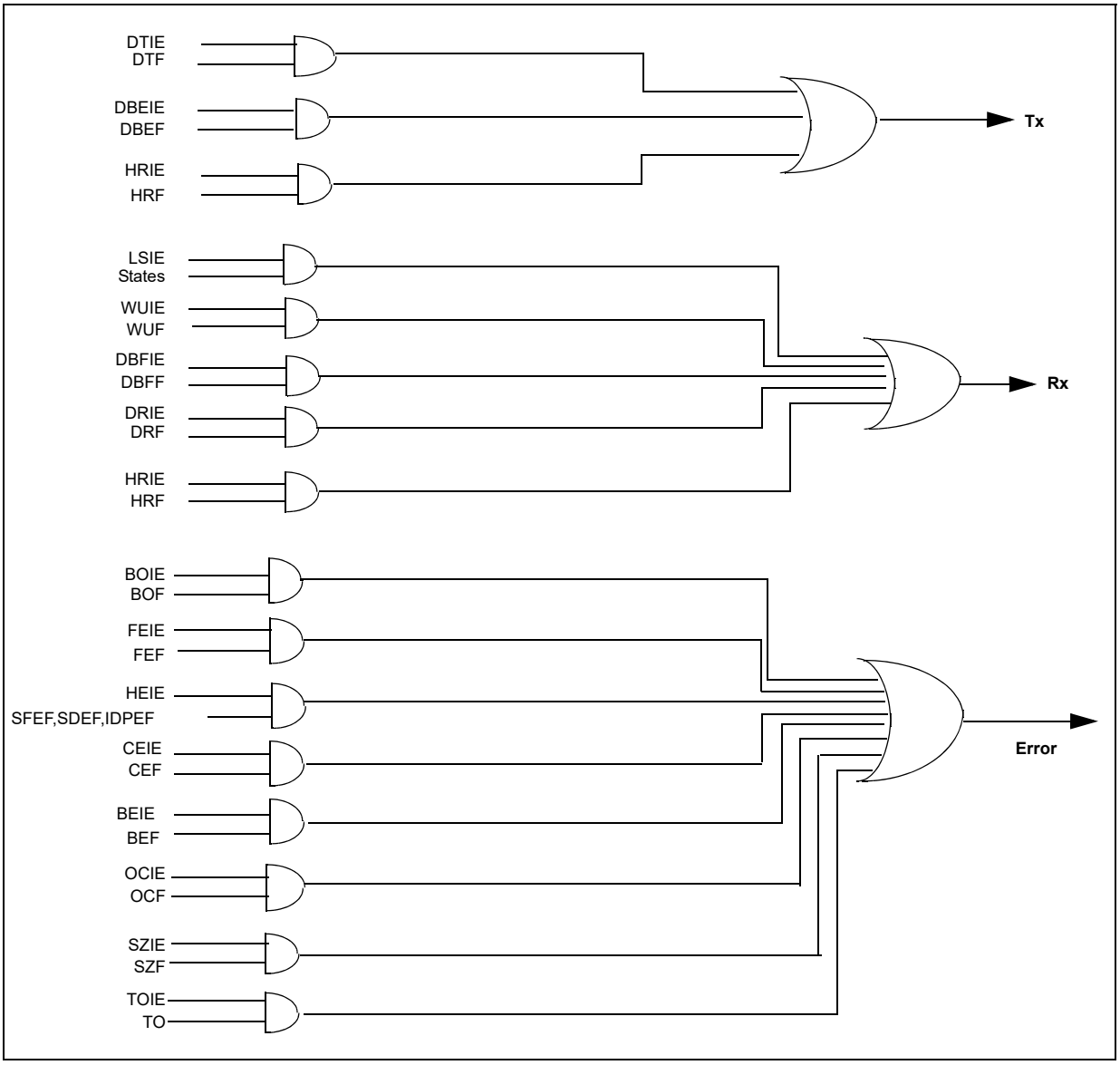
Table 873. Interrupts

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

Table 873. Interrupts(Continued)

Interrupt event	Event flag	Enable control bit	Interrupt vector
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
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 910. Interrupt diagram



51 Reset Generation Module (MC_RGM)

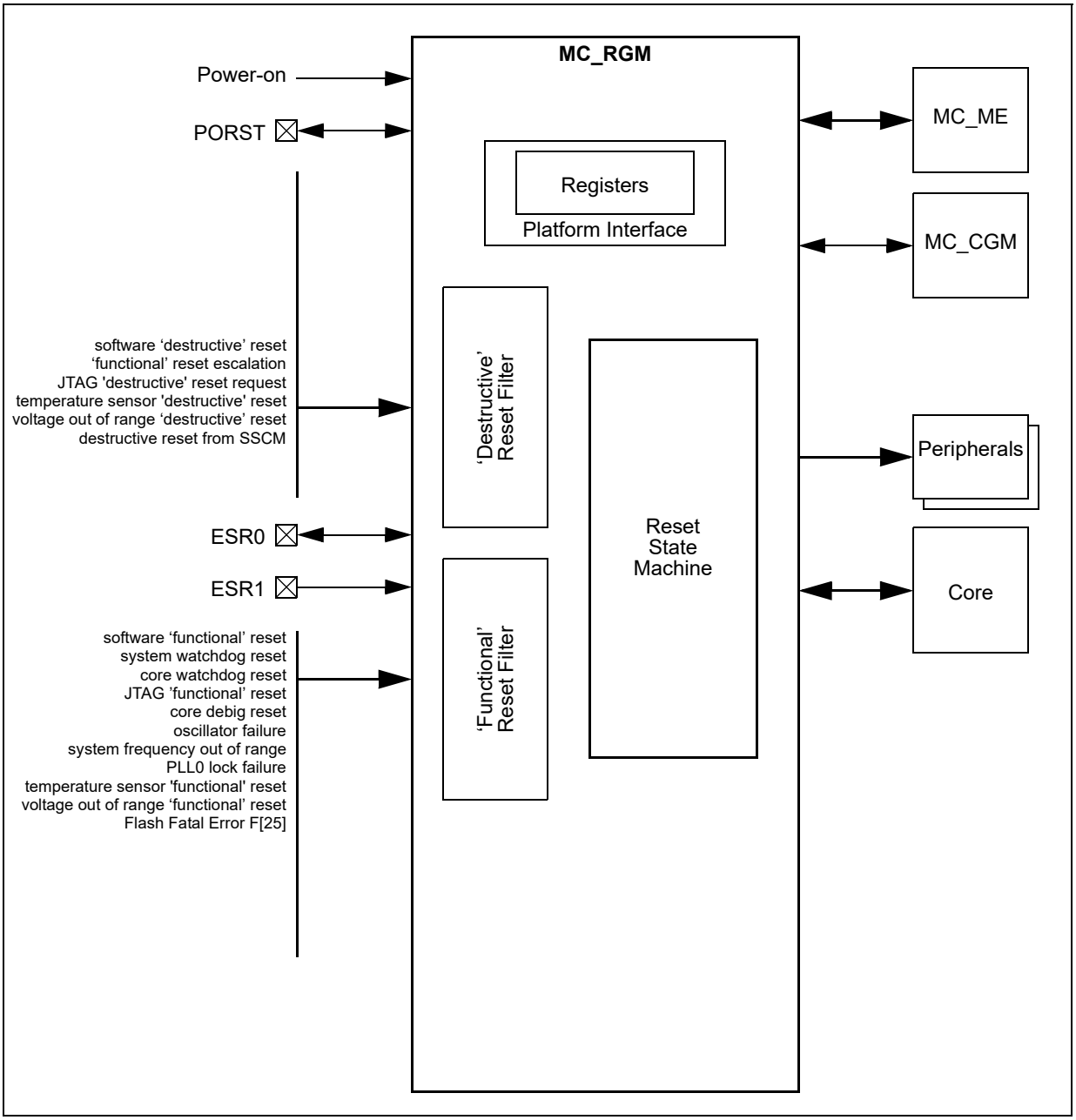
51.1 Introduction

51.1.1 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 911 shows the MC_RGM block diagram.

Figure 911. MC_RGM block diagram



51.1.2 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
- bidirectional reset behavior configuration
- configurable escalation of recurring ‘functional’ resets to ‘destructive’ reset

51.1.3 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
  – ’functional’ reset escalation
  – JTAG ’destructive’ reset request
  – SSCM ’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 (e.g., debug modules, flash modules) 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:

- ESR0 external reset
- ESR1 external reset
- software ’functional’ reset
- System Watchdog reset
- Core Watchdog reset
- JTAG ’functional’ reset request
- core debug reset
- oscillator failure
- system frequency out of range
- PLL0 lock failure
- temperature sensor ’functional’ reset
- voltage out of range ’functional’ reset
- Flash fatal error reset

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (i.e., 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 51.3.1.2: ‘Destructive’ Event Reset Disable (RGM_DERD) register* and *Section 51.3.1.3: ‘Destructive’ Event Alternate Request (RGM_DEAR) register* for ‘destructive’ resets and *Section 51.3.1.6: ‘Functional’ Event Reset Disable (RGM_FERD) register* and *Section 51.3.1.7: ‘Functional’ Event Alternate Request Register (RGM_FEAR)* for ‘functional’ resets).

51.2 External signal description

The MC_RGM interfaces to the bidirectional reset pin ESR0.

51.3 Memory map and register definition

Table 874. MC_RGM register description

Address	Register	Size	Access			Location
			User	Supervisor	Test	
0x000	'Destructive' Event Status Register (RGM_DES)	word	read	read/write(1)	read/write (1)	Section 51.3.1.1
0x010	'Destructive' Event Reset Disable Register (RGM_DERD)	word	read	read/write(2)	read/write (2)	Section 51.3.1.2
0x020	'Destructive' Event Alternate Request Register (RGM_DEAR)	word	read	read/write	read/write	Section 51.3.1.3
0x030	'Destructive' Bidirectional Reset Enable Register (RGM_DBRE)	word	read	read/write	read/write	Section 51.3.1.4
0x300	'Functional' Event Status Register (RGM_FES)	word	read	read/write(1)	read/write (1)	Section 51.3.1.5
0x310	'Functional' Event Reset Disable Register (RGM_FERD)	word	read	read/write(2)	read/write (2)	Section 51.3.1.6
0x320	'Functional' Event Alternate Request Register (RGM_FEAR)	word	read	read/write	read/write	Section 51.3.1.7
0x330	'Functional' Bidirectional Reset Enable Register (RGM_FBRE)	word	read	read/write	read/write	Section 51.3.1.8
0x340	'Functional' Event Short Sequence Register (RGM_FESS)	word	read	read/write	read/write	Section 51.3.1.9
0x604	'Functional' Reset Threshold Register (RGM_FRET)	byte	read	read/write	read/write	Section 51.3.1.10
0x60C	External Reset Output Assertion Control Register (RGM_EROEC)	byte	read	read/write	read/write	Section 51.3.1.11
0x610	Peripheral Reset 0 Register (RGM_PRST0)	word	read	read/write	read/write	Section 51.3.1.12
0x614	Peripheral Reset 1 Register (RGM_PRST1)	word	read	read/write	read/write	Section 51.3.1.13
0x618	Peripheral Reset 2 Register (RGM_PRST2)	word	read	read/write	read/write	Section 51.3.1.14
0x61C	Peripheral Reset 3 Register (RGM_PRST3)	word	read	read/write	read/write	Section 51.3.1.15
0x620	Peripheral Reset 4 Register (RGM_PRST4)	word	read	read/write	read/write	Section 51.3.1.16

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*

51.3.1 Register descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the 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.

51.3.1.1 ‘Destructive’ Event Status (RGM_DES) register

This register contains the status of the ‘destructive’ 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’. 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_DEST	0	0	0	F_JTAG_DEST	0	F_EDR	0	0	0	0	F_SOF_DEST	0	F_PORST	F_POR
W						w1c		w1c					w1c		w1c	w1c
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 912. ‘Destructive’ Event Status (RGM_DES) register

Table 875. ‘Destructive’ Event Status (RGM_DES) register Field Descriptions

Field	Description
0–6	Reserved
7 F_VOR_DEST	Flag for voltage out of range ‘destructive’ reset 0 No voltage out of range ‘destructive’ reset event has occurred since the last clear 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
9–16	Reserved

Table 875. ‘Destructive’ Event Status (RGM_DES) register Field Descriptions(Continued)

Field	Description
17 F_SSCM_DEST	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
18–20	Reserved
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	Reserved
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
24–27	Reserved
28 F_SOFT_DEST	Flag for software ‘destructive’ reset 0 No software ‘destructive’ reset event has occurred since the last clear 1 A software ‘destructive’ reset event has occurred
29	Reserved
30 F_PORST	Flag for external power-on reset 0 No external power-on reset has occurred since the last clear 1 A external power-on reset event has occurred
31 F_POR	Flag for Power-On reset 0 No power-on event has occurred since the last clear 1 A power-on event has occurred

51.3.1.2 ‘Destructive’ Event Reset Disable (RGM_DERD) register

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 51.3.1.3: ‘Destructive’ Event Alternate Request (RGM_DEAR) register*). 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.

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	0	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	0	0	0	0	0	0	0	0	0	0	0	0	0	D_PORST	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 913. ‘Destructive’ Event Reset Disable (RGM_DERD) register

Table 876. ‘Destructive’ Event Reset Disable (RGM_DERD) register Field Descriptions

Field	Description
0-7	Reserved
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 values in RGM_DEAR
9–29	Reserved
30 D_PORST	Disable external power-on reset 0 An external power-on reset event triggers a reset sequence 1 An external power-on reset event generates either a SAFE mode or an interrupt request depending on the values in RGM_DEAR
31	Reserved

51.3.1.3 ‘Destructive’ Event Alternate Request (RGM_DEAR) register

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 914. ‘Destructive’ Event Alternate Request (RGM_DEAR) register

Table 877. ‘Destructive’ Event Alternate Request (RGM_DEAR) register Field Descriptions

Field	Description
0–7	Reserved
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
9–29	Reserved
30 AR_PORST	Alternate Request for external power-on reset 0 Generate a SAFE mode request on an external power-on reset event if the reset is disabled 1 Generate an interrupt request on an external power-on reset event if the reset is disabled
31	Reserved

51.3.1.4 Destructive’ Bidirectional Reset Enable (RGM_DBRE) register

This register enables the generation of an external reset on ‘destructive’ 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.

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_DEST	0	0	0	BE_JTAG_DEST	0	BE_EDR	0	0	0	0	BE_SOF_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 915. Destructive' Bidirectional Reset Enable (RGM_DBRE) register

Table 878. Destructive' Bidirectional Reset Enable (RGM_DBRE) register Field Descriptions

Field	Description
0–6	Reserved
7 BE_VOR_DEST	Bidirectional Reset Enable for voltage out of range 'destructive' reset 0 ESR0 is asserted on a voltage out of range 'destructive' reset event
8 BE_TSR_DEST	Bidirectional Reset Enable for temperature sensor 'destructive' reset 0 ESR0 is asserted on a temperature sensor 'destructive' reset event if the reset is enabled
9–16	Reserved
17 BE_SSCM_DEST	Bidirectional Reset Enable for SSCM 'destructive' reset request 0 ESR0 is asserted on an SSCM 'destructive' reset request event
18–20	Reserved
21 BE_JTAG_DEST	Bidirectional Reset Enable for JTAG 'destructive' reset request 0 ESR0 is asserted on a JTAG 'destructive' reset request event 1 ESR0 is not asserted on a JTAG 'destructive' reset request event
22	Reserved
23 BE_EDR	Bidirectional Reset Enable for 'functional' reset escalation 0 ESR0 is asserted on a 'functional' reset escalation event
24–27	Reserved
28 BE_SOFT_DEST	Bidirectional Reset Enable for software 'destructive' reset 0 ESR0 is asserted on a software 'destructive' reset event
29	Reserved

Table 878. Destructive' Bidirectional Reset Enable (RGM_DBRE) register Field Descriptions

Field	Description
30 BE_PORST	Bidirectional Reset Enable for external power-on reset 0 ESR0 is asserted on an external power-on reset event if the reset is enabled
31 BE_POR	Bidirectional Reset Enable for Power-On reset 0 ESR0 is asserted on a Power-On reset event

51.3.1.5 'Functional' Event Status (RGM_FES) register

This register contains the status of the 'functional' reset sources, including those configured not to 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 has been properly cleared, and if not, clear it again.

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	F_FLASH	F_VOR_FUNC	F_TSR_FUNC	0	0	0	0	0	0	F_PLL0
W							w1c	w1c	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	F_CMU_FHL0	F_CMU_OLR	F_CORE	F_JTAG_FUNC	F_CWD	F_SWT	0	0	0	0	F_SOFT_FUNC	0	F_ESR1	F_ESR0
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	0

Figure 916. 'Functional' Event Status (RGM_FES) register

Table 879. ‘Functional’ Event Status (RGM_FES) register Field Descriptions

Field	Description
0–5	Reserved
6 F_FLASH	Flag for Flash fatal error reset 0 No Flash fatal error reset event has occurred since either the last clear or the last power-on reset assertion. 1 A Flash fatal error reset event has occurred.
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
9–14	Reserved
15 F_PLL0	Flag for PLL0 lock failure 0 No PLL0 lock failure event has occurred since either the last clear or the last power-on reset assertion. 1 A PLL0 lock failure event has occurred.
16–17	Reserved
18 F_CMU_FHL0	Flag for system frequency out of range 0 No system frequency out of range event has occurred since either the last clear or the last power-on reset assertion. 1 A system frequency out of range event has occurred.
19 F_CMU_OLR	Flag for oscillator failure 0 No oscillator failure event has occurred since either the last clear or the last power-on reset assertion. 1 A oscillator failure event has occurred.
20 F_CORE	Flag for core debug reset 0 No core debug reset event has occurred since either the last clear or the last power-on reset assertion. 1 A core debug reset event has occurred.
21 F_JTAG_FUNC	Flag for JTAG ’functional’ reset request 0 No JTAG ’functional’ reset request event has occurred since either the last clear or the last power-on reset assertion. 1 A JTAG ’functional’ reset request event has occurred.
22 F_CWD	Flag for Core Watchdog reset 0 No Core Watchdog reset event has occurred since either the last clear or the last power-on reset assertion. 1 A Core Watchdog reset event has occurred.
23 F_SWT	Flag for System Watchdog reset 0 No System Watchdog reset event has occurred since either the last clear or the last power-on reset assertion. 1 A System Watchdog reset event has occurred.
24–27	Reserved

Table 879. 'Functional' Event Status (RGM_FES) register Field Descriptions

Field	Description
28 F_SOFT_FUNC	Flag for software 'functional' reset 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	Reserved
30 F_ESR1	Flag for ESR1 External Reset 0 No ESR1 external reset event has occurred since either the last clear or the last power-on reset assertion 1 An ESR1 external reset event has occurred
31 F_ESR0	Flag for ESR0 External Reset 0 No ESR0 external reset event has occurred since either the last clear or the last power-on reset assertion 1 An ESR0 external reset event has occurred

51.3.1.6 'Functional' Event Reset Disable (RGM_FERD) register

This register provides dedicated bits to disable 'functional' reset sources.When a 'functional' reset source is disabled, the associated 'functional' event triggers either a SAFE mode request or an interrupt request (see *Section 51.3.1.7: '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	D_FLASH	D_VOR_FUNC	D_TSR_FUNC	0	0	0	0	0	0	D_PLL0
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	D_CMU_FHL0	D_CMU_OLR	D_CORE	0	D_CWD	D_SWT	0	0	0	0	0	0	D_ESR1	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 917. 'Functional' Event Reset Disable (RGM_FERD) register

Table 880. 'Functional' Event Reset Disable (RGM_FERD) register field descriptions

Field	Description
0–5	Reserved
6 D_FLASH	Disable Flash fatal error reset 0 A Flash fatal error reset event triggers a reset sequence 1 A Flash fatal error reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_FLASH
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_JTAG_FUNC
9–14	Reserved
15 D_PLL0	Disable PLL0 lock failure 0 A PLL0 lock failure event triggers a reset sequence 1 A PLL0 lock failure event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_PLL0
16–17	Reserved
18 D_CMU_FHL0	Disable system frequency out of range 0 A system frequency out of range event triggers a reset sequence 1 A system frequency out of range event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_CMU_FHL0

Table 880. 'Functional' Event Reset Disable (RGM_FERD) register field descriptions(Continued)

Field	Description
19 D_CMU_OLR	Disable oscillator failure 0 A oscillator failure event triggers a reset sequence 1 A oscillator failure event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_CMU_OLR
20 D_CORE	Disable core debug reset 0 A core debug reset event triggers a reset sequence 1 A core debug reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_CORE
21	Reserved
22 D_CWD	Disable Core Watchdog reset 0 A Core Watchdog reset event triggers a reset sequence 1 A Core Watchdog reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_CWD
23 D_SWT	Disable System Watchdog reset 0 A System Watchdog reset event triggers a reset sequence 1 A System Watchdog reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_SWT
24–29	Reserved
30 D_ESR1	Disable ESR1 External Reset 0 An ESR1 external reset event triggers a reset sequence 1 An ESR1 external reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_ESR1
31	Reserved

51.3.1.7 '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

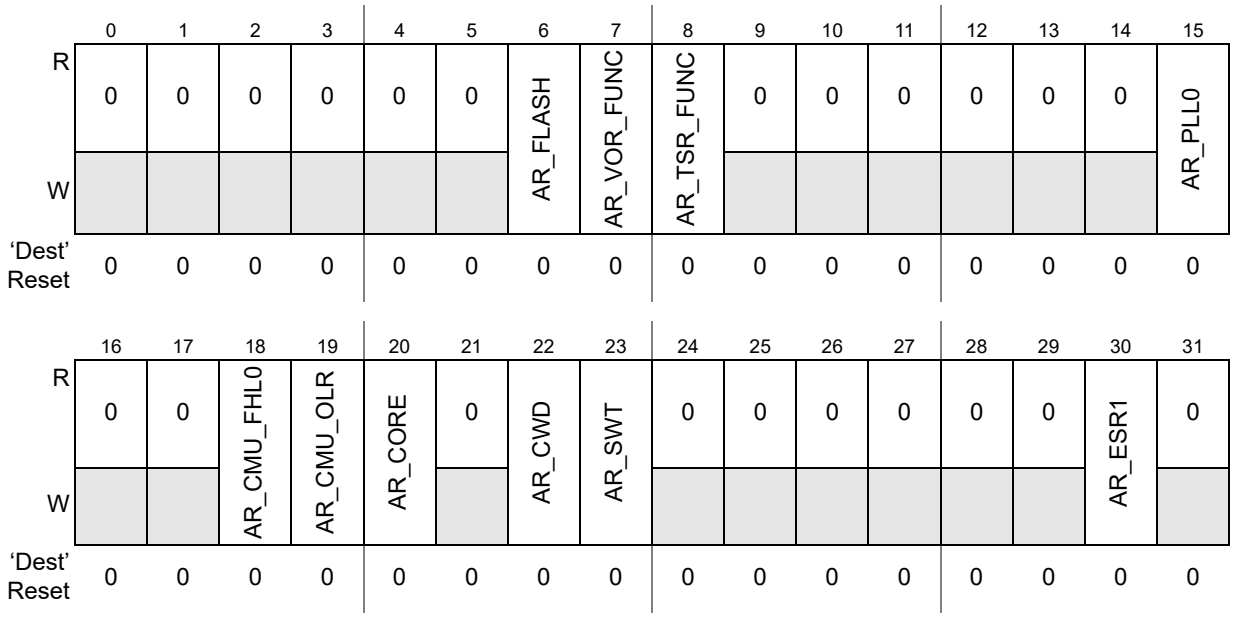


Figure 918. 'Functional' Event Alternate Request Register (RGM_FEAR)

Table 881. 'Functional' Event Alternate Request Register (RGM_FEAR) field descriptions

Field	Description
0–5	Reserved
6 AR_FLASH	Alternate Request for Flash fatal error reset 0 Generate a SAFE mode request on a Flash fatal error reset event if the reset is disabled 1 Generate an interrupt request on a Flash fatal error reset event if the reset is disabled
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
9–14	Reserved
15 AR_PLL0	Alternate Request for PLL0 lock failure 0 Generate a SAFE mode request on a PLL0 lock failure event if the reset is disabled 1 Generate an interrupt request on a PLL0 lock failure event if the reset is disabled
16–17	Reserved

Table 881. 'Functional' Event Alternate Request Register (RGM_FEAR) field descriptions(Continued)

Field	Description
18 AR_CMU_FHL0	Alternate Request for system frequency out of range 0 Generate a SAFE mode request on a system frequency out of range event if the reset is disabled 1 Generate an interrupt request on a system frequency out of range event if the reset is disabled
19 AR_CMU_OLR	Alternate Request for oscillator failure 0 Generate a SAFE mode request on a oscillator failure event if the reset is disabled 1 Generate an interrupt request on a oscillator failure event if the reset is disabled
20 AR_CORE	Alternate Request for core debug reset 0 Generate a SAFE mode request on a core debug reset event if the reset is disabled 1 Generate an interrupt request on a core debug reset event if the reset is disabled
21	Reserved
22 AR_CWD	Alternate Request for Core Watchdog reset 0 Generate a SAFE mode request on a Core Watchdog reset event if the reset is disabled 1 Generate an interrupt request on a Core Watchdog reset event if the reset is disabled
23 AR_SWT	Alternate Request for System Watchdog reset 0 Generate a SAFE mode request on a System Watchdog reset event if the reset is disabled 1 Generate an interrupt request on a System Watchdog reset event if the reset is disabled
24–29	Reserved
30 AR_ESR1	Alternate Request for ESR1 External Reset 0 Generate a SAFE mode request on an ESR1 external reset event if the reset is disabled 1 Generate an interrupt request on an ESR1 external reset event if the reset is disabled
31	Reserved

51.3.1.8 'Functional' Bidirectional Reset Enable (RGM_FBRE) register

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	BE_FLASH	BE_VOR_FUNC	BE_TSR_FUNC	0	0	0	0	0	0	BE_PLL0
W																
'Dest' Reset	0	0	0	0	0	0	0	1	1	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	BE_CMU_FHL0	BE_CMU_OLR	BE_CORE	BE_JTAG_FUNC	BE_CWD	BE_SWT	0	0	0	0	BE_SOFT_FUNC	0	BE_ESR1	BE_ESR0
W																
'Dest' Reset	0	0	1	1	1	1	1	1	0	0	0	0	1	0	1	0

Figure 919. 'Functional' Bidirectional Reset Enable (RGM_FBRE) register

Table 882. 'Functional' Bidirectional Reset Enable (RGM_FBRE) register field descriptions

Field	Description
0–5	Reserved
6 BE_FLASH	Bidirectional reset enable for Flash fatal error reset 0 ESR0 is asserted on a Flash fatal error reset event if the reset is enabled 1 ESR0 is not asserted on a Flash fatal error reset event
7 BE_VOR_FUNC	Bidirectional Reset Enable for voltage out of range 'functional' reset 0 ESR0 is asserted on a voltage out of range 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a voltage out of range 'functional' reset event
8 BE_TSR_FUNC	Bidirectional Reset Enable for temperature sensor 'functional' reset 0 ESR0 is asserted on a temperature sensor 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a temperature sensor 'functional' reset event
9–14	Reserved
15 BE_PLL0	Bidirectional Reset Enable for PLL0 'functional' reset 0 ESR0 is asserted on a PLL0 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a PLL0 'functional' reset event
16–17	Reserved
18 BE_CMU_FHL0	Bidirectional reset enable for system frequency out of range 0 ESR0 is asserted on a system frequency out of range event if the reset is enabled 1 ESR0 is not asserted on a system frequency out of range event
19 BE_CMU_OLR	Bidirectional reset enable for oscillator failure 0 ESR0 is asserted on an oscillator failure event if the reset is enabled 1 ESR0 is not asserted on an oscillator failure event

Table 882. 'Functional' Bidirectional Reset Enable (RGM_FBRE) register field descriptions(Continued)

Field	Description
20 BE_CORE	Bidirectional reset enable for core debug reset 0 ESR0 is asserted on a core debug reset event if the reset is enabled 1 ESR0 is not asserted on a core debug reset event
21 BE_JTAG_FUNC	Bidirectional reset enable for JTAG 'functional' reset request 0 ESR0 is asserted on a JTAG 'functional' reset request event if the reset is enabled 1 ESR0 is not asserted on a JTAG 'functional' reset request event
22 BE_CWD	Bidirectional reset enable for Core Watchdog reset 0 ESR0 is asserted on a Core Watchdog reset event if the reset is enabled 1 ESR0 is not asserted on a Core Watchdog reset event
23 BE_SWT	Bidirectional reset enable for System Watchdog reset 0 ESR0 is asserted on a System Watchdog reset event if the reset is enabled 1 ESR0 is not asserted on a System Watchdog reset event
24–27	Reserved
28 BE_SOFT_FUNC	Bidirectional Reset Enable for software 'functional' reset 0 ESR0 is asserted on a software 'functional' reset event if the reset is enabled 1 ESR0 is not asserted on a software 'functional' reset event
29	Reserved
30 BE_ESR1	Bidirectional Reset Enable for ESR1 External Reset 0 ESR0 is asserted on an ESR1 external reset event if the reset is enabled 1 ESR0 is not asserted on an ESR1 external reset event
31 BE_ESR0	Bidirectional Reset Enable for ESR0 External Reset 0 ESR0 is asserted on an ESR0 external reset event if the reset is enabled

51.3.1.9 'Functional' Event Short Sequence (RGM_FESS) register

This register defines which reset sequence is followed when a 'functional' reset sequence is triggered. The 'functional' reset sequence can either start from PHASE1 or from PHASE3, skipping PHASE1 and PHASE2. This is useful when a fast reset sequence is desired, skipping flash reset, for example.

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	SS_FLASH	SS_VOR_FUNC	SS_TSR_FUNC	0	0	0	0	0	0	SS_PLL0
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	SS_CMU_FHL0	SS_CMU_OLR	SS_CORE	SS_JTAG_FUNC	SS_CWD	SS_SWT	0	0	0	0	SS_SOFT_FUNC	0	SS_ESR0	SS_ESR0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 920. 'Functional' Event Short Sequence (RGM_FESS) register

Table 883. 'Functional' Event Short Sequence (RGM_FESS) register field descriptions

Field	Description
0–5	Reserved
6 SS_FLASH	Short Sequence for Flash fatal error reset 0 The reset sequence triggered by a Flash fatal error reset event starts from PHASE1 1 The reset sequence triggered by a Flash fatal error reset event starts from PHASE3
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 will start from PHASE1 1 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
8 SS_TSR_FUNC	Short Sequence for temperature sensor 'functional' reset 0 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
9–14	Reserved
15 SS_PLL0	Short Sequence for PLL0 lock failure 0 The reset sequence triggered by a PLL0 lock failure event starts from PHASE1 1 The reset sequence triggered by a PLL0 lock failure event starts from PHASE3
16–17	Reserved
18 SS_CMU_FHL0	Short Sequence for system frequency out of range 0 The reset sequence triggered by a system frequency out of range event starts from PHASE1 1 The reset sequence triggered by a system frequency out of range event starts from PHASE3

Table 883. 'Functional' Event Short Sequence (RGM_FESS) register field descriptions(Continued)

Field	Description
19 SS_CMU_OLR	Short Sequence for oscillator failure 0 The reset sequence triggered by a oscillator failure event starts from PHASE1 1 The reset sequence triggered by a oscillator failure event starts from PHASE3
20 SS_CORE	Short Sequence for core debug reset 0 The reset sequence triggered by a core debug reset event starts from PHASE1 1 The reset sequence triggered by a core debug reset event starts from PHASE3
21 SS_JTAG_FUNC	Short Sequence for JTAG 'functional' reset 0 The reset sequence triggered by a JTAG 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a JTAG 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
22 SS_CWD	Short Sequence for Core Watchdog reset 0 The reset sequence triggered by a Core Watchdog reset event starts from PHASE1 1 The reset sequence triggered by a Core Watchdog reset event starts from PHASE3
23 SS_SWT	Short Sequence for System Watchdog reset 0 The reset sequence triggered by a System Watchdog reset event starts from PHASE1 1 The reset sequence triggered by a System Watchdog reset event starts from PHASE3
24–27	Reserved
28 SS_SOFT_FUNC	Short Sequence for software 'functional' reset 0 The reset sequence triggered by a software 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a software 'functional' reset event will start from PHASE3, skipping PHASE1 and PHASE2
29	Reserved
30 SS_ESR1	Short Sequence for ESR1 External Reset 0 The reset sequence triggered by an ESR1 external reset event will start from PHASE1 1 The reset sequence triggered by an ESR1 external reset event if the reset is enabled will start from PHASE3, skipping PHASE1 and PHASE2
31 SS_ESR0	Short Sequence for ESR0 External Reset 0 The reset sequence triggered by an ESR0 external reset event will start from PHASE1 1 The reset sequence triggered by an ESR0 external reset event if the reset is enabled will start from PHASE3, skipping PHASE1 and PHASE2

51.3.1.10 'Functional' Reset Escalation Threshold (RGM_FRET) register

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 enables the 'functional' reset escalation function. Writing any value to this register resets the 'functional' reset counter. See *Section 51.4.6: 'Functional' reset escalation*, for details on the 'functional' reset escalation function.

This register is reset only on power-on and any 'destructive' reset.

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 921. 'Functional' Reset Escalation Threshold (RGM_FRET) register

Table 884. 'Functional' Reset Escalation Threshold (RGM_FRET) register field descriptions

Field	Description
0–3	Reserved
4–7 FRET	'Functional' Reset Escalation Threshold If the value of this field is 0, the 'functional' reset escalation function is disabled. Any other value is the number of 'functional' resets which cause a 'destructive' reset if the RGM_FRET register is not written to beforehand.

51.3.1.11 External Reset Output Extension Control (RGM_EROEC) register

This register controls the assertion of the ESR0 pin after the reset sequence has completed. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. The register bit is cleared on write '1'.

Clearing the EROEC bit causes the MC_RGM to stop asserting the ESR0 pin within two 16 MHz internal RC oscillator clock cycles. The ESR0 pin cannot be asserted by writing to this register.

This register also provides the current value sampled on ESR0 via the ERIS bit. This is useful for software to poll if, for example, ESR0 is used in the system to synchronize the application software with an external event.

This register is reset only on power-on.

Address 0x60C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	ERIS	EROEC
W								w1c
POR	0	0	0	0	0	0	0	1

Figure 922. External Reset Output Extension Control (RGM_EROEC) register

Table 885. External Reset Output Extension Control (RGM_EROEC) register field descriptions

Field	Description
0–5	Reserved
6 ERIS	External Reset Input Status 0 A ‘0’ is being observed on ESR0 1 A ‘1’ is being observed on ESR0
7 EROEC	External Reset Output Assertion Control 0 ESR0 is not being asserted by the MC_RGM 1 ESR0 is being asserted by the MC_RGM

51.3.1.12 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.

Caution: Use the **RGM_PRST*0*** register with care. See *Section 51.4.7: Individual peripheral resets*, for the proper sequence to prevent unexpected chip behavior.

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	PIT_RTC_1_RST	PIT_RTC_0_RST	DECIFILTER_RST	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	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	SIPI_0_RST	0	LFAST_0_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 923. Peripheral Reset Register 0 (RGM_PRST0)

Table 886. Peripheral Reset Registers (RGM_PRST*0*) Field Descriptions

Field	Description
0 PIT_RTC_1_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
1 PIT_RTC_0_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
2 DECFILTER_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
3–15	Reserved
16 SIUL_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
17–19	Reserved
20 SIPI_0_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
21	Reserved
22 LFAST_0_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
23–31	Reserved

51.3.1.13 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.

Caution: Use the **RGM_PRST*1*** register with care. See *Section 51.4.7: Individual peripheral resets*, for the proper sequence to prevent unexpected chip behavior.

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	ADCSD_2_RST	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	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 924. Peripheral Reset Register 1 (RGM_PRST1)

Table 887. Peripheral Reset Registers (RGM_PRST*1*) Field Descriptions

Field	Description
0–3	Reserved
4 ADCSD_2_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
5–26	Reserved
27 DMAMUX_0_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
28-31	Reserved

51.3.1.14 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.

Caution: Use the **RGM_PRST*2*** register with care. See *Section 51.4.7: Individual peripheral resets*, for the proper sequence to prevent unexpected chip behavior.

Address 0x618 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LINFlexD_0_RST	LINFlexD_1_RST	0	0	0	0	0	LINFlexD_14_RST	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CAN_RAM_CTR_RST	CCCU_RST	0	0	MCAN_1_RST	MCAN_2_RST	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. Peripheral Reset Register 2 (RGM_PRST2)

Table 888. Peripheral Reset Registers (RGM_PRST2) Field Descriptions

Field	Description
0–2	Reserved
3 LINFlexD_0_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
4 LINFlexD_1_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
5–9	Reserved
10 LINFlexD_14_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
11–20	Reserved

Table 888. Peripheral Reset Registers (RGM_PRST*2*) Field Descriptions(Continued)

Field	Description
21 CAN_RAM_CTR_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
22 CCCU_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
23–24	Reserved
25 MCAN_1_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
26 MCAN_2_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
27–31	Reserved

51.3.1.15 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.

Caution: Use the **RGM_PRST*3*** register with care. See *Section 51.4.7: Individual peripheral resets*, for the proper sequence to prevent unexpected chip behavior.

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_0_RST	0	0	0	ADCSAR_4_RST	0	0	0	0	0	0	0	0	0	0	ADCSAR_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	SENT_0_RST	0	0	0	0	DSPI_0_RST	0	DSPI_4_RST	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 926. Peripheral Reset Register 3 (RGM_PRST3)

Table 889. Peripheral Reset Registers (RGM_PRST*3*) Field Descriptions

Field	Description
0 ADCSAR_0_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
1–3	Reserved
4 ADCSAR_4_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
5–14	Reserved
15 ADCSAR_b_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
16–22	Reserved
23 SENT_0_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
24–27	Reserved

Table 889. Peripheral Reset Registers (RGM_PRST*3*) Field Descriptions(Continued)

Field	Description
28 DSPI_0_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
29	Reserved
30 DSPI_4_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral
31	Reserved

51.3.1.16 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.

Caution: Use the **RGM_PRST*4*** register with care. See *Section 51.4.7: Individual peripheral resets*, for the proper sequence to prevent unexpected chip behavior.

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	0	0	0	
W																GTMINT_RST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 927. Peripheral Reset Register 4 (RGM_PRST4)

Table 890. Peripheral Reset Registers (RGM_PRST*4*) Field Descriptions

Field	Description
0–30	Reserved
31 GTMINT_RST	Peripheral reset — Writing a ‘1’ to this bit will reset the corresponding peripheral. Writing a ‘0’ to this bit will release the reset to the corresponding peripheral if the bit’s current value is ‘1’, otherwise it will have no effect. 0 no forced reset of peripheral 1 forced reset of peripheral

51.4 Functional description

51.4.1 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 891*.

Table 891. MC_RGM reset implications

Source	What Gets Reset	External Reset Assertion(1)	Boot Mode Capture
Power-on reset	All	Yes	Yes
‘Destructive’ resets	All except some clock/reset management	Yes	Yes
External reset	All except some clock/reset management and debug	Programmable (2)	Yes
‘Functional’ resets	All except some clock/reset management and debug	Programmable (2)	Programmable (3)
Shortened ‘functional’ resets(4)	Flip-flops except some clock/reset management and debug	Programmable (2)	Programmable (3)

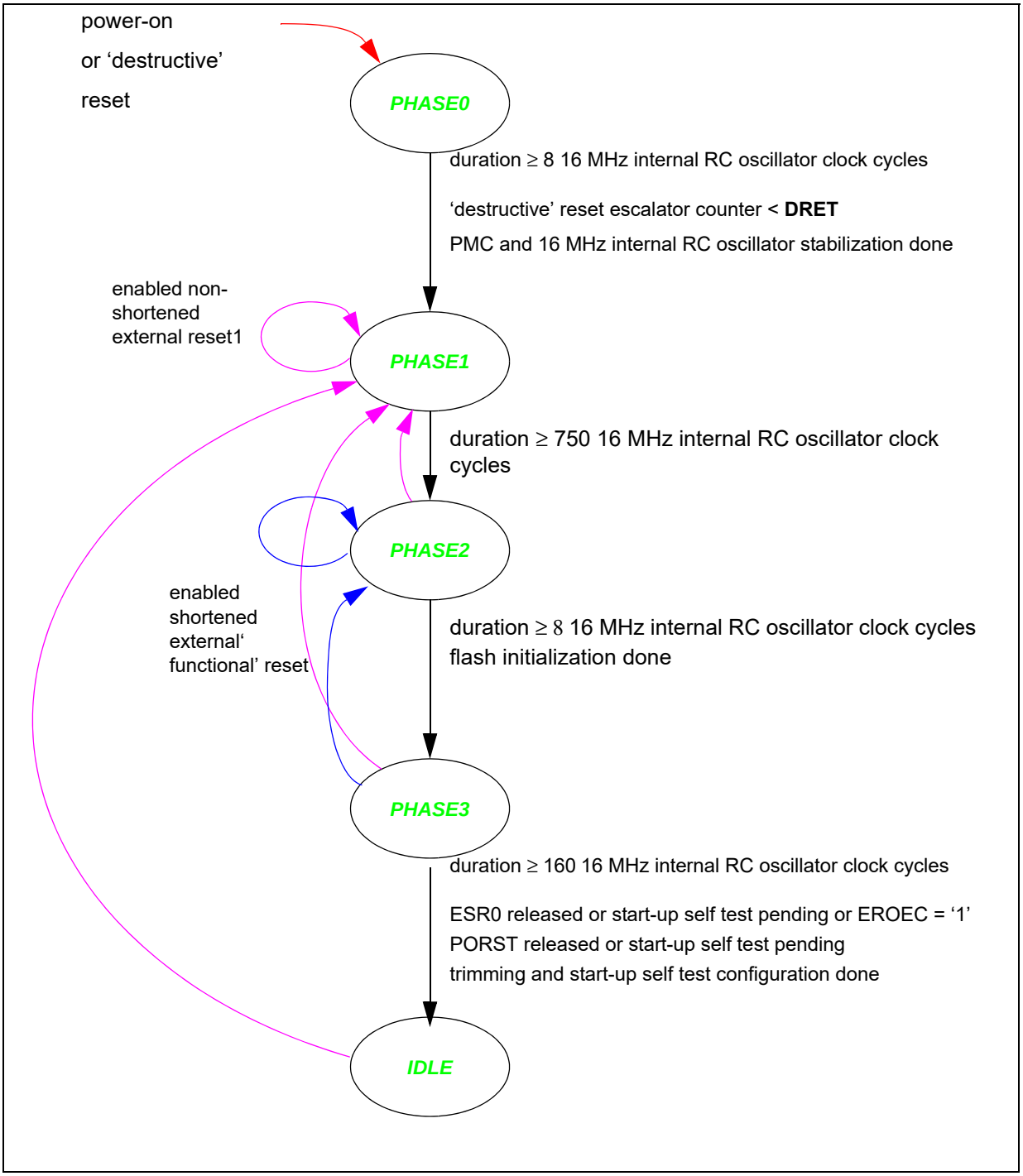
1. ‘external reset assertion’ means that the ESR0 pin is asserted by the MC_RGM from the start of the reset sequence until the end of reset PHASE3
2. the assertion of the external reset is controlled via the RGM_FBRE register
3. the boot mode is captured if the external reset is asserted
4. the short 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 928*.

Figure 928. MC_RGM State Machine



51.4.1.1 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
- a minimum of 8 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 resister

51.4.1.2 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 750 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled external or non-shortened ‘functional’ reset event

51.4.1.3 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 have completed flash initialization
- a minimum of 8 16 MHz internal RC oscillator clock cycles have elapsed since entering PHASE2

51.4.1.4 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 and start-up self test configuration
- a minimum of 160 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled, shortened ‘functional’ reset event
- ESR0 is not asserted, except if a start-up self test is to be executed on exit of the current reset sequence or the EROEC bit of the RGM_EROEC register is ‘1’
- PORST is not asserted, except if a start-up self test is to be executed on exit of the current reset sequence

51.4.1.5 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.

51.4.2 ‘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 is 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 triggers a reset sequence starting from the beginning of PHASE0.

51.4.2.1 External power-on reset—PORST

The bidirectional external power-on reset pin PORST is a special case ‘destructive’ reset.

The detection of a falling edge on PORST starts the reset sequence.RGM_PHASE1

The chip asserts PORST if the reset sequence was triggered by one of the following:

- a power-on reset
- a ‘destructive’ reset event

In this case, PORST is asserted until the end of PHASE0.

The PORST input is disabled immediately as of the PORST output being asserted in order to prevent a falling edge from being detected while the pin is being driven from the chip. The input is then re-enabled 4 ms after the PORST output stops being driven by the chip in order to allow the pull-up on the pin to take effect.

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.

51.4.3 External reset

The MC_RGM manages the external reset coming from ESR0. The detection of a falling edge on ESR0 starts the reset sequence.

The status flag associated with the external reset falling edge event (RGM_FES.F_ESR0 bit) is set when the external reset is asserted and the power-on reset is not asserted.

The external reset can optionally be disabled by writing bit RGM_FERD.D_ESR0.

Note: The RGM_FERD register can be written only once between two 'destructive' reset events.

An enabled external reset normally triggers a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by the external reset. When RGM_FESS.SS_ESR0 is set, the external reset triggers a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially when an external reset should not reset the flash.

The MC_RGM may also assert the external reset if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event
- an external reset event, if configured via the RGM_FBRE register to assert the external reset
- a 'functional' reset event configured via the RGM_FBRE register to assert the external reset

In this case, ESR0 is asserted until all conditions for exiting PHASE3 have been met with the exception of the ESR0 assertion check.

If the OPT_RGM_ESR0_HW bit of the "UTEST Miscellaneous" DCF client is set to '0', the MC_RGM continues to assert ESR0 until the EROEC bit in the RGM_EROEC register has been cleared by software.

In addition, the MC_RGM asserts ESR0 while the start-up self test is being executed.

The ESR0 input is disabled immediately as of the ESR0 output being asserted in order to prevent a falling edge from being detected while the pin is being driven from the chip. The input is then re-enabled 4 µs after the ESR0 output stops being driven by the chip in order to allow the pull-up on the pin to take effect.

If ESR0 is stopped being asserted by the MC_RGM and the 4 µs input mask period expires during the reset IDLE phase (e.g., due to the EROEC bit being cleared by software), and now a low value is detected on ESR0, an external reset falling edge event does not occur, and no new reset sequence is triggered. For a new reset sequence to trigger, ESR0 must first be deasserted.

51.4.4 '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 normally triggers 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 triggers 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.

51.4.5 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 892*.

Table 892. MC_RGM Alternate Event Selection

RGM_D/FERD bit value	RGM_D/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 (i.e., at the reset source that caused the alternate request) and also clearing the appropriate RGM_D/FES status bit.

- *SAFE mode requests are generated regardless of whether the system clock is running. Interrupt requests are generated with the system clock, and therefore, if the system clock was disabled at the time of the event, the interrupt request is not asserted until the system clock is re-enabled.RGM_TIMING*
- *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 will 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 does not send any safe mode/interrupt request to the MC_ME.*

51.4.6 ‘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 (i.e., 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.

51.4.7 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.

52 Boot Assist Flash (BAF)

52.1 Introduction

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 boot core (Main Processor on this 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. The BAF does not execute further since the read access to flash memory boot sectors for the core may be disabled if device life cycle is FAIL_ANALYSIS. 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. The package pins used by LINFlexD are the same pins used by the M_CAN module. Hence, the external PHY can be either LIN or CAN, allowing the “ASC@CAN” operation (see *Section 52.2.6, Serial boot configuration* for configuration details).

52.1.1 BAF image header

BAF image contains a 256-byte header starting from address 0040_4000h (BAF version number, see the table “BAF memory map” in the “BAF Configuration”). The header contains important information such as BAF version. The BAF header is explained in *Table 893*.

Table 893. BAF image header

Address offset	Size (bits)	Field description
0000h	32	BAF Image version
0004h	32	Reserved
0008h	32	Reserved (Part of clock Jitter Activation Constant)
000Ch	32	Clock Jitter Activation Constant
0010h–00FFh	Reserved	

52.1.1.1 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 894. 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
[16:23]	Reserved
[24:31]	Reserved

52.1.1.2 Clock Jitter Activation constant

The Clock Jitter Activation constant is a 32-bit flag, but the smallest element in flash that can be programmed is 64 bits, so the first 32 bits are reserved to allow programming of this constant. This 32-bit flag can be programmed by the customer and is used to activate the Clock Jitter feature that adds clock edge jitter to the baud rate clocks of several communications modules. As part of the BAF startup procedure, BAF copies this constant from flash to the PASS Clock Jitter Enable Register.

Table 895. BAF Clock Jitter Activation Constant

Field	Description
[00:31]	Clock Jitter Activation Constant. The initial state of production disable flag is erased - FFFF_FFFFh. This equates to NO clock jitter.

52.2 Functional description

52.2.1 Boot modes

Depending on the state and the content of its flash memory, the device can enter one of three different boot modes.

- Boot from internal flash

  If the internal flash contains application code with a valid boot header, the application is booted. The boot header contains flags that enable/disable individual cores and their reset vectors.
- Serial boot

  If no valid boot header is found and the current life cycle phase of the device is less than IN_FIELD, an attempt is made to download the application by a serial protocol using the LINFlexD or M_CAN module. The downloaded code is then executed by the initial boot core.
- Static mode

  The device enters power safe mode and waits for an external reset.

52.2.2 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:

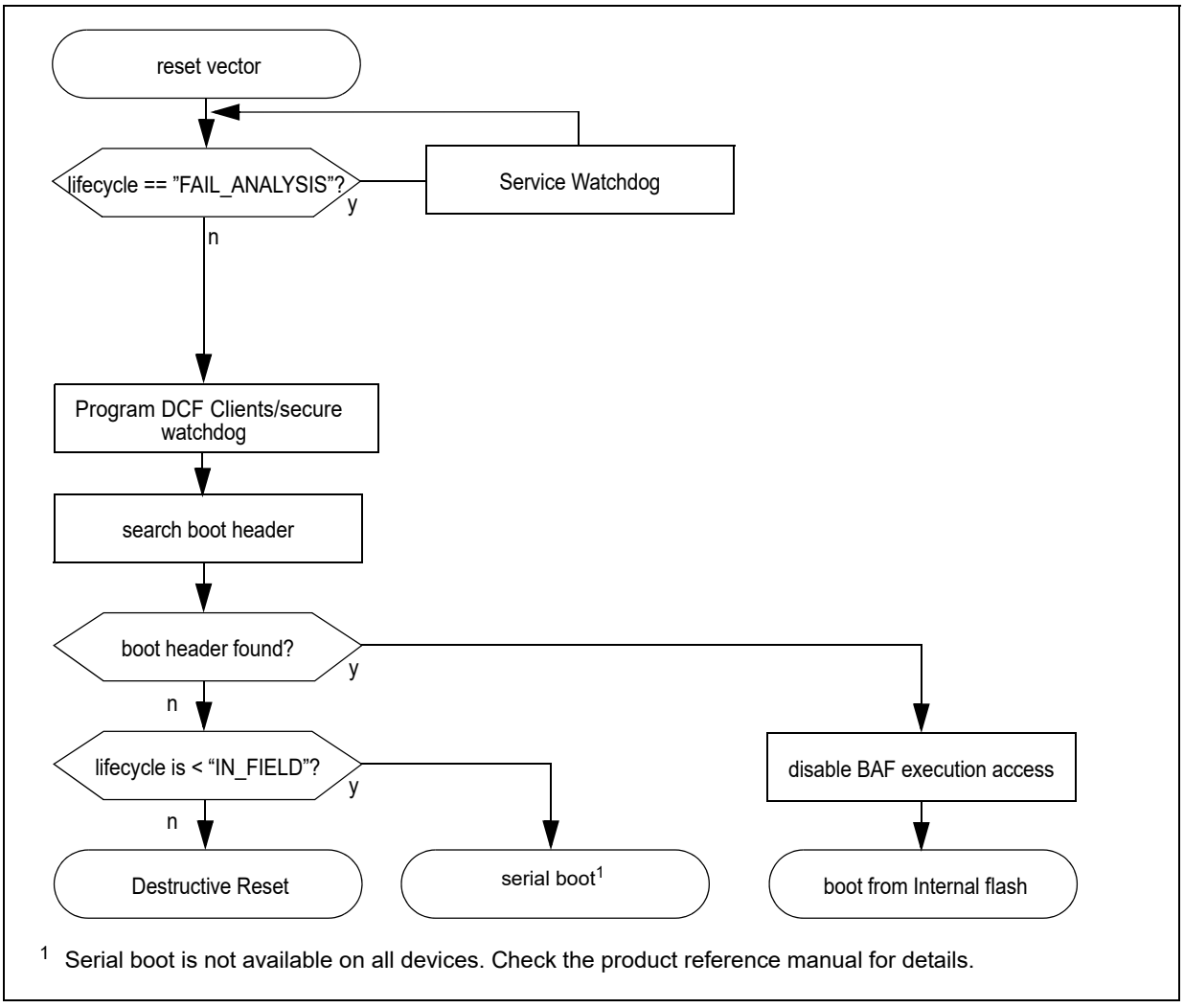
1. BAF sets up the exception vector table to catch machine check exceptions, which may occur if there are any uncorrectable ECC errors. In the event of an uncorrectable ECC error, the BAF issues a destructive reset from the machine check exception handler if the exception is due to the reading of instructions 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.
2. BAF uses memory of the system RAM as in the "BAF memory map" table (in *Section 6.9.1: BAF configuration*) 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.

52.2.3 Flow of control

Figure 929 shows the overall flow of control in the BAF. The individual steps are explained in the following sections.

Figure 929. Flow of control



1 Serial boot is not available on all devices. Check the product reference manual for details.

52.2.4 Initialization of BAF DCF clients

52.2.4.1 '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 896*.

The DCF clients which are used for initialization of security watchdog are explained in *Section 52.2.4.2, Security watchdog configuration*.

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 896* 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 anywhere in the CPU address space including the BAF code block or UTEST. 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 52.2.4.6, SER_BOOT_CBACK – BAF serial boot DCF client*.

Table 896. BAF DCF client list

Address (hex)	Name	Label	Description
0000h	Security Watchdog Control Register	SEC_SWT_CR	Data to write to SWT Control Register
0001h	Security watchdog Time out register	SEC_SWT_TO	Security watchdog counter value
0002h	Security watchdog service address	SEC_IAC8	Instruction Address compare register value - Code present at this address services the watchdog.
0003h	Security watchdog CPU select	SEC_SWT_CPU	Determines which CPU is to use the security watchdog
0004h–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–7FFFh	Reserved	—	Reserved for future use.

52.2.4.2 Security watchdog configuration

BAF initializes the security watchdog before it jumps to the HOST CPU application. BAF implements four DCF clients for security watchdog initialization. BAF parses the DCF records and looks for configuration data to be used to program the security watchdog registers. BAF first collects all information required to program the security watchdog from DCF records and then program the security watchdog registers. BAF configures only SWT2 as security watchdog for the Main core. Security watchdog detailed configuration steps are explained in the section below.

52.2.4.2.1 Security Watchdog Control Register (SEC_SWT_CR)

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0000h while parsing, BAF programs the DCF client “Security watchdog control register” with the 32-bit value present in the DATA section of the DCF records. Once written, any subsequent writes to this DCF Client are ignored by BAF.

52.2.4.2.2 Security Watchdog TImeout (SEC_SWT_TO)

The BAF parses the DCF records. While parsing, if the BAF finds a DCF record with CS14 set and address field as 0001h, the BAF programs the security watchdog timeout DCF Clients. This is the security watchdog timeout register. The BAF checks the value obtained from DCF records. If the value yields a timeout greater than 30 seconds, then the BAF

changes this value to 30 seconds. Once written, any subsequent writes to this DCF client are ignored.

52.2.4.2.3 Security Watchdog Service Address (SEC_IAC8)

BAF configures the security watchdog in ‘fixed address execution service mode’. In this mode, the watchdog is serviced by executing code at the address loaded into the designated IAC8 (Instruction Address Compare) register. If BAF finds a DCF record with CS14 set and address field as 0002h while parsing the DCF records, BAF programs the DCF clients for SEC_IAC8. This DCF client can only be written once. Subsequent writes are ignored.

Warning: To program the core IAC8 special purpose register, the ownership of this register should be with software. If hardware has its ownership software will not be able to write to it using mtspr instruction.

52.2.4.2.4 Security Watchdog CPU select (SEC_SWT_CPU)

This DCF client indicates to BAF which CPU will use the security watchdog. BAF parses the DCF record and if it finds a record with CS14 set and address field set as 0003h, BAF programs this DCF client. This means that BAF uses the two least significant bits out of the 32-bit data in the DCF record to find the CPU number which will use the security watchdog. *Figure 930* and *Table 897* indicate the data layout for this client in the DCF record. This client is written once during reset. Subsequent writes are ignored.

Writing this client causes the BAF to configure the security watchdog based on data written to the three previous DCF clients. If any of the three previous DCF clients are not written, data is configured erroneously.

If security watchdog is assigned to CPU0, relevant CPU IAC8 register is programmed along with other debug configuration registers to enable IAC8 as watch point.

During normal BAF operation, SWT2 is configured to perform a conventional task watchdog function (cause reset if not regularly serviced). To assign the security watchdog to the Main CPU, first SWT3 is configured to perform the watchdog function task. The timeout value, configuration, and current time is read from SWT2 and copied to SWT3. Then SWT2 is unlocked with current sequence and enabled as security watchdog.

DCF Client Address 0003h

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

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				0				0				0	0	SC1	SC0

Figure 930. SEC_SWT_CPU bit configuration

Table 897. SEC_SWT_CPU field description

Field	Description
[31:2]	Reserved
SC[1:0]	These two bits select the CPU which will use the security watchdog. 00 Reserved 01 Reserved 10 Main CPU 11 Reserved

52.2.4.3 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.

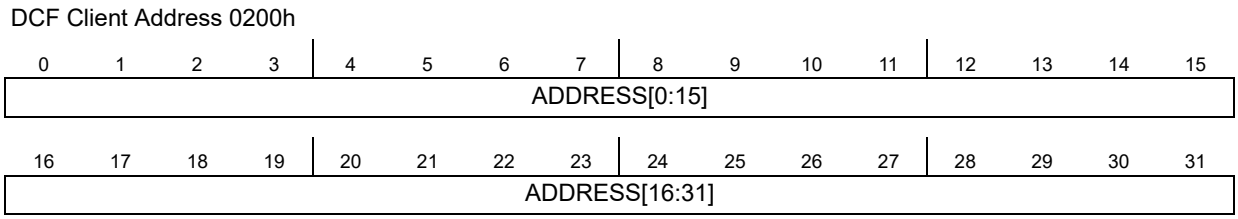


Figure 931. BAF DCF ADDR client configuration

Table 898. BAF DCF ADDR client description

Field	Description
ADDR[0:31]	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.

52.2.4.4 BAF DCF client - DATAX

52.2.4.4.1 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.

DCF Client Address 0201h

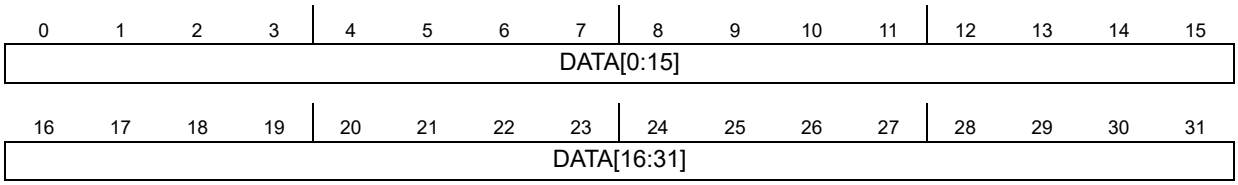


Figure 932. DATA32 DCF client configuration

52.2.4.4.2 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.

DCF Client Address 0202h

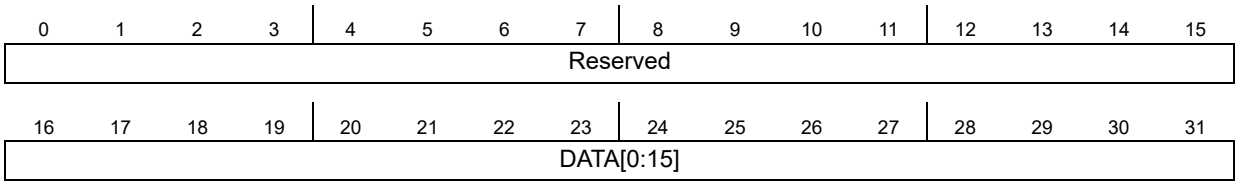


Figure 933. DATA16 DCF Client Configuration

52.2.4.4.3 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.

DCF Client Address 0203h

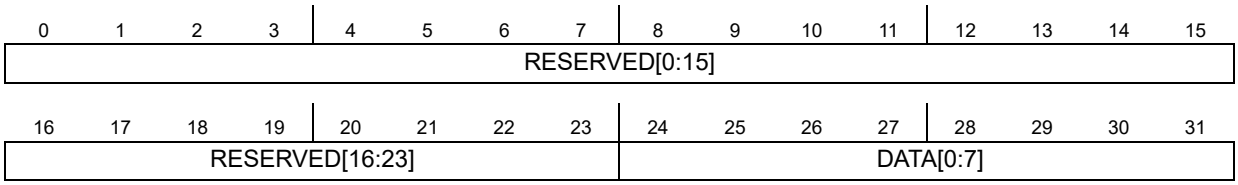


Figure 934. DATA8 DCF Client Configuration

52.2.4.5 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 power.org EABI 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.

52.2.4.6 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 power.org EABI 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 899* for details regarding return value).

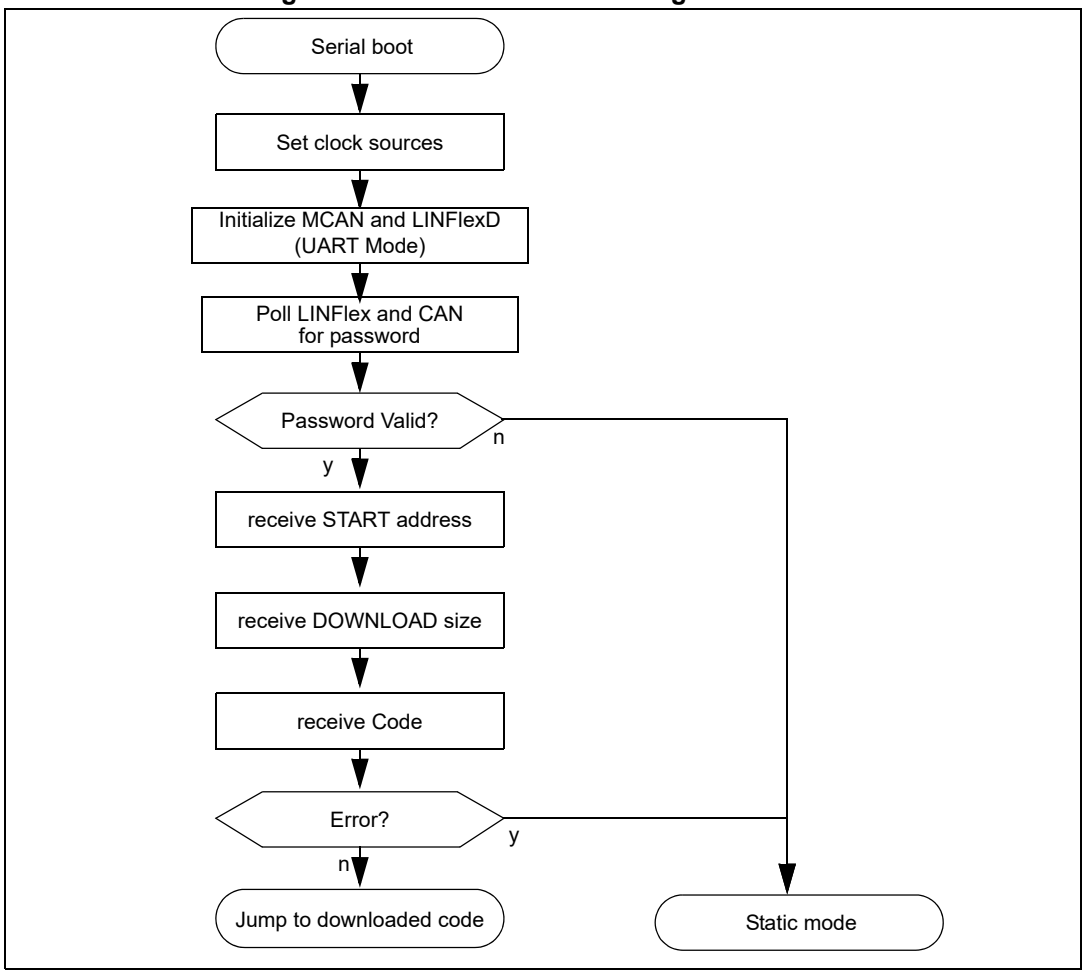
Table 899. Serial boot callback function return value definition

Return values	Description
00h	Boot from internal flash using standard search for boot record (see ‘Serial boot header and boot options’ in configuration section).
01h	Serial boot using both M_CAN and LINFlexD (in UART mode) (see ‘Serial boot header and boot options’ in configuration section).
02h–FFh	Reserved

52.2.5 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 in CUST_DELIV (Customer Delivery), it attempts a serial boot. Otherwise, the boot has failed and the BAF issues a destructive reset.

Figure 935. Flow of control during serial boot



The clock source in the MC_CGM is configured and the PADs are set up for LINFlexD and M_CAN signals (see *Section 52.2.6, Serial boot configuration*), then the code is downloaded using LINFlexD or M_CAN in Serial Boot mode. In the absence of errors causing the device to be placed in Static mode, the device state is restored to resemble the out-of-reset state and the downloaded code is executed.

Serial boot allows the download of application to internal sram via LINFlexD or M_CAN interface. After downloading the application image to SRAM, the BAF jumps to the start address provided as part of download protocol mentioned below and executes the downloaded application.

At the start of the process both LINFlexD and M_CAN is 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 does not participate further in serial boot.

If a CAN message of 8 bytes with ID 11h is received on the M_CAN controller first, the BAF program

- Disables LINFlexD.
- Configures LINFlexD RXD input mux back to reset state.
- Transitions to CAN serial download protocol routine.

If CAN message not containing 8 bytes or NOT with ID 11h is received by M_CAN controller or a M_CAN controller generates a CAN error, it is probable that the tool is sending a LINFlexD frame. Under such circumstances BAF program

- Ignores the CAN message, if a message arrived.
- Clear any pending CAN error
- Continue monitoring for valid CAN messages.

If any byte from LINFlexD is received first, the BAF program:

- Checks if the first byte is 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 M_CAN controller and restore to its reset state.
- Configures the M_CAN pads to their reset state.
- Transition to LINFlexD serial boot (UART mode).

52.2.5.1 Serial boot mode protocol

From a high-level perspective, the download protocol over CAN and LIN interfaces follows the steps below:

1. Host transmits the 64-bit password to MCU.
2. Host transmits start address, size of downloaded code in bytes, NO_ECHO bit and VLE bit to MCU.
3. Host transmits a sequence of data bytes that are to be executed.
4. CPU executes code from start address provided in step 2.

Each step must be complete before the next step starts.

The exact protocol between LINFlexD and MCAN are slightly different.

NO_ECHO bit is used to select if the actual download data are to be echoed or not. If NO_ECHO bit is set, then the data will not be echoed otherwise if NO_ECHO bit is clear then the data will be echoed.

52.2.5.2 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).

52.2.5.3 MCAN download protocol

The MCAN download protocol uses very similar structure to the LINFlexD protocol, but it makes use of the message ID's provided by CAN.

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 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.

52.2.5.4 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.

52.2.5.5 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 bits, NO_ECHO bit and 30-bit code length as shown in *Figure 936* and *Figure 937*

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.

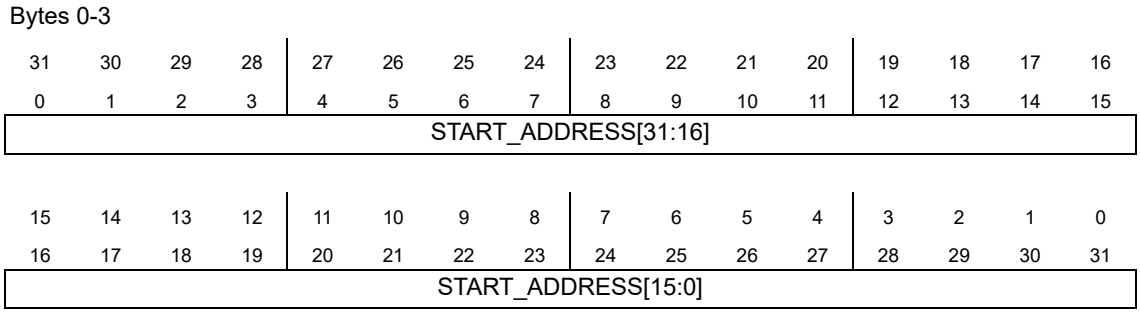


Figure 936. Start address, VLE bit, NO_ECHO bit and download size in bytes

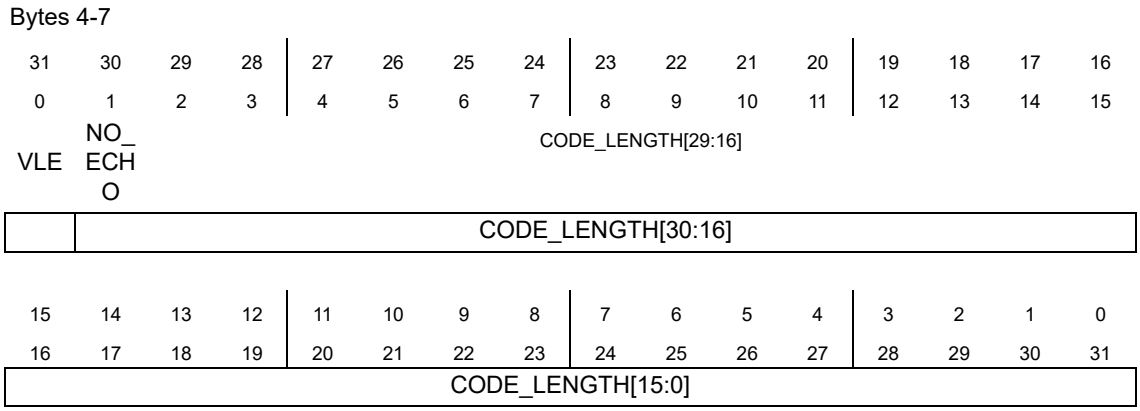


Figure 937. Start address, VLE bit, NO_ECHO bit and download size in bytes

52.2.5.6 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.

52.2.5.7 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.

52.2.6 Serial boot configuration

52.2.6.1 LINFlexD configuration

Boot using UART protocol is implemented by LINFlexD module.

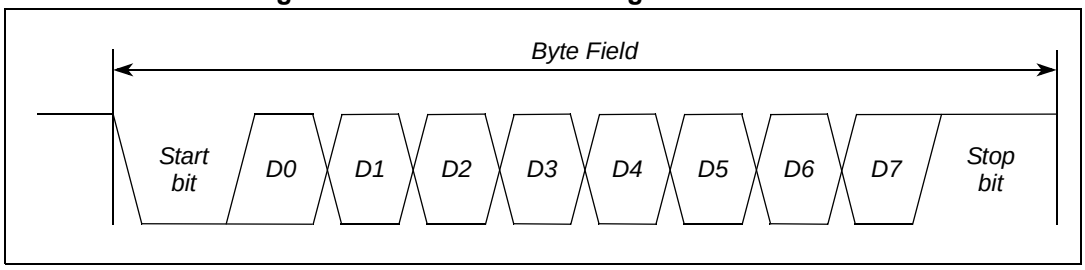
Note: See the pin configuration table in the configuration for pins used by the LINFlexD module.

When Serial Boot mode is started the LINFlexD_Rx is configured as an input and multiplexed to the LINFlexD_0 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 (such as LIN or CAN). 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 at a baud rate of 250 kbaud with the IRCOSC as reference clock, using an 8-bit data frame without parity bit and 1 stop bit.

Figure 938. LINFlexD bit timing in UART mode

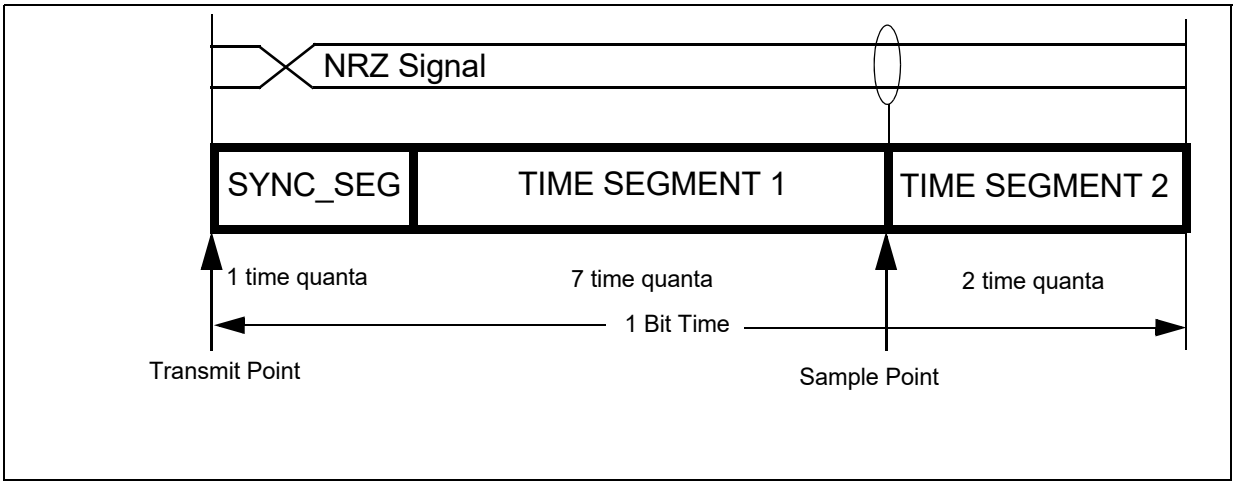


52.2.6.2 M_CAN configuration

The M_CAN controller is configured to operate at a baud rate equal to crystal frequency divided by 40, using the standard 11-bit identifier format detailed in CAN 2.0A specification (see *Table 900* for baud rates).

The Bit timing is configured as shown in *Figure 939 (M_CAN Bit Timing)*

Figure 939. M_CAN Bit Timing



52.2.6.3 Baud rates

The LINFlexD baud rate is set to 250 kbaud with the IRCOSC as reference clock.

The M_CAN baud rate is set by the frequency of the external crystal divided by 40. *Table 900* shows some possible baud rates.

Table 900. M_CAN_1 baud rates

Crystal frequency	M_CAN baud rate
8 MHz	200 kbaud
12 MHz	300 kbaud
16 MHz	400 kbaud
20 MHz	500 kbaud
40 MHz	1000 kbaud

52.2.6.4 LINFlexD and M_CAN pin configuration

See the BAF configuration section for pin details.

52.2.6.5 Protocol summary

Table 901 summarizes the LINFlexD protocol and BAF action during this boot mode.

Table 901. 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 901. 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 echoed, otherwise data is not 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 902 shows the M_CAN serial boot protocol.

Table 902. M_CAN 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

52.3 Resources

- Clocks

  Several clock configurations are made depending on the boot mode of the BAF. All settings are restored before the execution of the application code (in flash memory or received via serial download).
- LINFlexD and M_CAN

  The LINFlexD module is only used if normal boot header search fails and a serial download boot is attempted. All settings are restored before the application code in system RAM is executed.
- Interrupts

  No interrupts/exceptions are used or enabled, except the Machine Check exception, to handle ECC errors.
- Security watchdog

  When is chosen as security watchdog, SWT3 is used for task monitoring purpose.

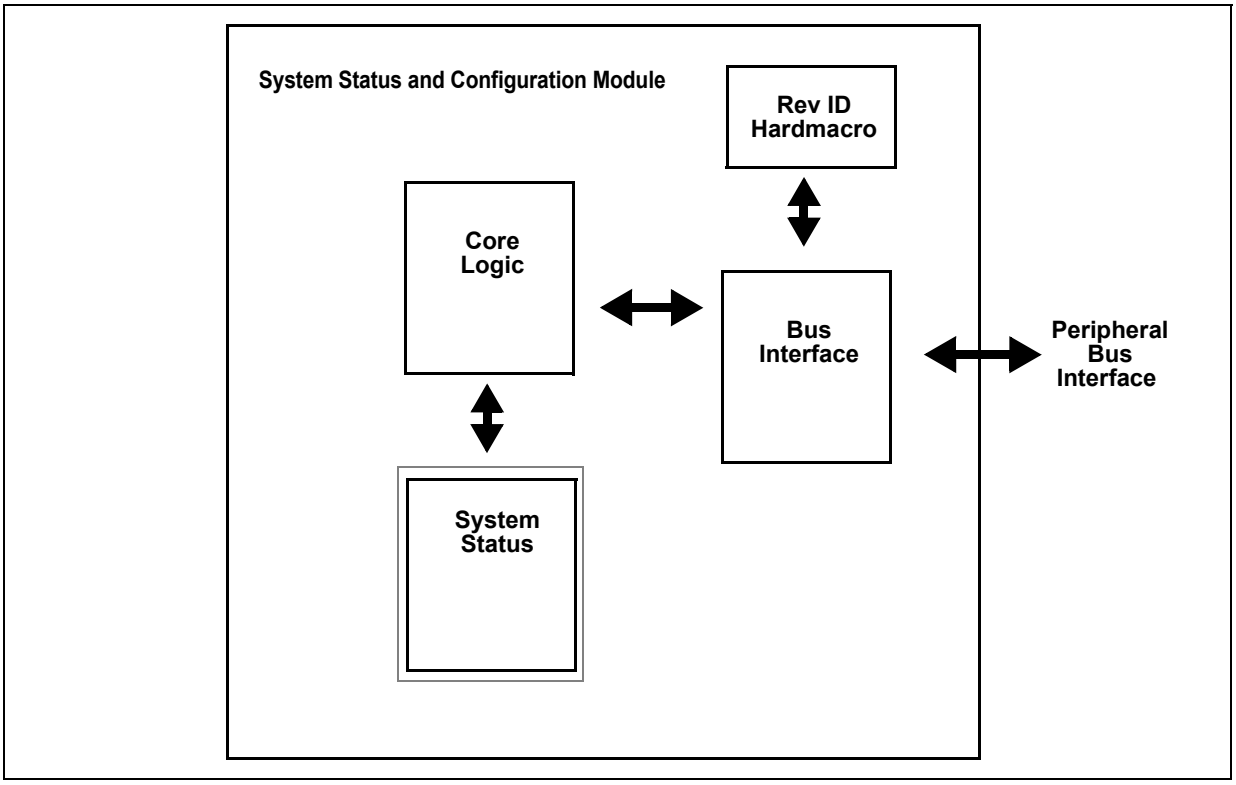
53 System Status and Configuration Module (SSCM)

53.1 Introduction

53.1.1 Overview

The System Status and Configuration Module (SSCM) is pictured in *Figure 940*.

Figure 940. SSCM block diagram



53.1.2 Glossary

Table 903. Glossary

Term	Definition
0011b	Binary numbers
0Fh	Hexadecimal numbers
Pin	External physical connection.
Signal	Electronic construct whose state or change in state conveys information.
X	In certain contexts, such as a signal encoding, this indicates a don’t care. For example, if a field is binary coded X001b, the state of the first bit is a don’t care.
y	y is used for a place holder of 1 hex digit for unknown values.

53.1.3 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

53.1.4 Modes of operation

The SSCM operates identically in all system modes.

53.2 External signal description

The SSCM has no external pins.

53.3 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

Table 904 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 + 0002h, but performing a 16-bit access to Base + 0003h is illegal.

Table 904. SSCM memory map

Offset from SSCM_BASE	Register	Access	Reset Value	Section/ page
0000h	SSCM System Status (STATUS)	R/W	yyy0h(1)	*on page 1526*
0002h	SSCM System Memory and ID Register (MEMCONFIG)	R	yyyyh(1)	*on page 1527*
0004h	Reserved	—	—	—
0006h	SSCM Error Configuration Register (ERROR)	R/W	0000h	*on page 1527*
0008h	Reserved(2)	—	—	—
000Ah–001Ch	Reserved	—	—	—
0020h	Reserved	—	—	—
0024h	Reserved	—	—	—
0028h	Processor Start Address Register (PSA)	R	yyyy_yyyyh(1)	*on page 1528*
002Ch	Reserved	—	—	—
0030h	Reserved	—	—	—
0034h	Life Cycle Status Register (LCSTAT)	R		

Table 904. SSCM memory map(Continued)

Offset from SSCM_BASE	Register	Access	Reset Value	Section/ page
0038h-0048h	Reserved	—	—	—

1. See register description for reset value details.
2. No register abort on this location.

53.3.1 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.

53.3.1.1 System Status Register (STATUS)

The System Status register reflects the current state of the system.

Offset 0000h Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CER	0	NXEN1	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 941. System Status Register (STATUS)

Table 905. STATUS field descriptions

Field	Description
CER 1	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.
NXEN1 3	Processor 1 Nexus enable status. 0Processor 1 Nexus disabled. 1Processor 1 Nexus enabled
NXEN 4	Processor 0 Nexus enable status. 0Processor 0 Nexus disabled. 1Processor 0 Nexus enabled

Table 905. STATUS field descriptions(Continued)

Field	Description
BMODE 8:10	Device Boot Mode. 101 Boot from BootFlash All other values are reserved. This field is only updated during reset.
VLE 11	Variable Length Instruction Mode. When booting in Single Chip (SC) 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. 0SC Boot Location contains standard PPC code 1SC Boot Location contains VLE code

53.3.1.2 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 0002h Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	JPIN										1	MREV				0
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	1	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0

1. Reset value reflects the JTAG ID of the device.

Figure 942. System memory and ID register (MEMCONFIG)

Table 906. MEMCONFIG field descriptions

Field	Description
JPIN 0:9	JTAG Part ID Number
MREV 11:14	Minor Mask Revision

53.3.1.3 Error Configuration Register (ERROR)

The Error Configuration register is a read-write register that controls the error handling of the system.

Offset 0006h 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 943. Error Configuration Register (ERROR)

Table 907. ERROR field descriptions

Field	Description
PAE 14	Peripheral Bus Abort Enable. This bit enables bus aborts on any access to an off-platform peripheral slot that is not used on the device. This feature is intended to aid in debugging when developing application code. 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
RAE 15	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 AIPS level). In this case, the PAE and RAE register bits have no effect on the abort.

53.3.1.4 Processor Start Address Register (PSA)

Offset 0028h 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 944. Processor Start Address Register (PSA)

Table 908. PSA field descriptions

Field	Description
SADR 0:31	Start Address - the boot processor will start executing application code from this address. In SC mode this indicates the location determined by the RCHW search. In BAF mode this will show the start address of the BAF code.

53.3.1.5 Life Cycle Status Register (LCSTAT)

Offset 0034h 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	Reserved		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)

Figure 945. Life Cycle Status Register (LCSTAT)

1. Reset value reflects the Pre-Life Cycle of the device.
2. Reset value is the Life Cycle of the device.

Table 909. LCSTAT field descriptions

Field	Description
LC 29:31	Life Cycle - as determined by the SSCM by reading the LC slots. The encoding is: JDP_PROD = 110 CUST_DELIV = 011 PROD_OEM = 010 IN_FIELD = 111 FAIL_ANALYSIS = 000 JDP_PRODUCTION = 110

53.4 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.

53.4.1 DCF mechanism

The DCF mechanism has been developed to handle settings of device parameters via OTP flash memory.

Starting at location UTEST_OFFSET the user can store a series of DCF records - the device will process these records during the system reset sequence before the CPU leaves reset. The first record needs to be a start record followed by an application-specific number of data records as required, and finally a stop record.

Each record is 64 bits in length. *Figure 946* 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 946. DCF start record

0:31
05AA 55AFh

0:31
0000 0000h

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 947*. In a data record the STOP bit must always be programmed to 0. Some clients support a parity bit.

Figure 947. Format of a DCF data record

0:31
WDATA[31:0]

0:14	15:29	30	31
CS[14:0]	ADDR[16:2]	PRTY	STOP

The stop record indicates that processing should stop. The general format of the stop record is shown in *Figure 948*. 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 948. DCF stop record

0:31
rsvd

0:30	31
rsvd	1

If *n* data records are to be stored in UTEST, the data structure must be as shown in *Table 910*.

Table 910. Series of DCF records in UTEST

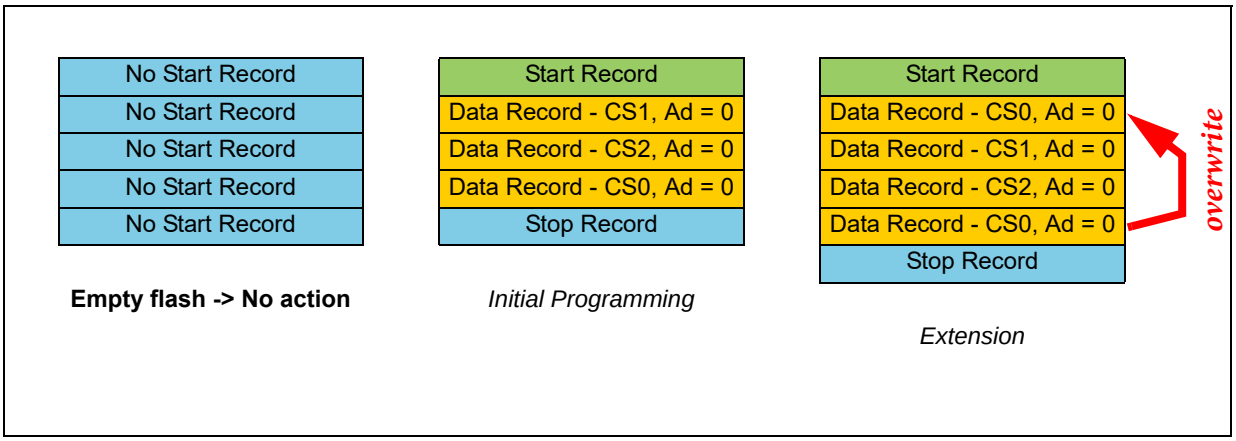
ADDR offset	DATA			
00h	05AA 55AFh			
04h	0000 0000h			STOP=0
08h	WDATA[31:0]			
0Ch	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
10h	WDATA[31:0]			
14h	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
8n - 1 + 0h	reserved			
8n - 1 + 4h	reserved			1

Table 910. Series of DCF records in UTEST(Continued)

ADDR offset	DATA
8n + 0h	
8n + 4h	

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 949*.

Figure 949. 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 0000_0001h then the PRTY field needs to be set to 1, so that the total number of 1s is even.)

53.4.2 ECC error monitoring

During flash memory scanning (such as Life Cycle, DCF records, or RCHW), 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 reset.

53.4.3 Boot mode functionality

The device supports the following boot modes for the main boot core:

- Boot Assisted by Flash (BAF) – the device will boot from UTest NVM Flash Block, (interface with the HSM if available) and perform device setup.

If booting is not possible with the selected configuration (for example, no Boot ID is found in the selected boot location) then the device will enter serial boot mode.

53.4.4 BAF configuration

By default the device will boot from BAF.

In BAF mode the boot core will start to execute the BAF code at the location shown in the SSCM section of the “Device Configuration” chapter. The BMODE field in the SSCM status register will be set to the value 101b. Details of the BAF operation are described in the BAF chapter.

53.4.5 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:

- JDP 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 statuses:

Table 911. Life Cycle slots

LC Slot		LC slot value
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 55AA_50AF_55AA_50AFh, “Erased” is detected by the bit pattern FFFF_FFFF_FFFF_FFFFh. *Table 912* shows how the slots are arranged in memory.

Table 912. Life Cycle Slots in Memory

Offset	LC Slot word
00h	Valid[31:0]
04h	Valid[63:32]

Table 912. Life Cycle Slots in Memory(Continued)

Offset	LC Slot word
08h	Invalid[31:0]
0Ch	Invalid[63:32]

The Life Cycle is determined as shown in *Table 913*. The priority of the entries is from top to bottom, so the first row which applies determines the resulting Life Cycle.

Table 913. Life Cycle Slots

LC Slot 0 JDP_PROD(1),(2)	LC Slot 1 CUST_DEL(1),(2)	LC Slot 2 OEM_PROD(1),(2)	LC Slot 3 IN_FIELD(1),(2)	LC Slot 4 FA(1),(2)	Resulting Life Cycle
Active	Erased	Erased	Erased	Erased	JDP 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
*	*	*	*	Erased	In Field
*	*	*	*	Not Erased(3)	System Reset

1. '*' = any other value.
2. 'Erased' = slot returns all 1's.
3. Any value other than FFFF_FFFF_FFFF_FFFFh.

53.5 Initialization and application information

53.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (see *Section 53.3.1, Register descriptions*).

53.6 Additional safety measures

53.6.1 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.

54 Power Management Controller digital interface (PMC_dig)

54.1 Introduction

The power management controller (PMC) consists of two blocks: an analog block and a digital block, which is described here.

The digital block:

- Contains the registers and digital logic required to generate the enable signals, trim control bits, POR reset generation and test logic for the analog block.
- Contains the controls for PMC analog outputs to be sensed with the ADC module.
- Contains the enable signals, trim bits, and other registers regarding the temperature sensor.
- Interfaces to the Reset Generation Module (MC_RGM) block, which receives the LVD values during POR to use for phase transition conditions.
- Has a custom interface to Flash memory via the SSCM (and DCF client blocks) which loads the trim values during initial POR.

Caution: Do not assert the TESTPIN or the system LVDs will be deactivated.

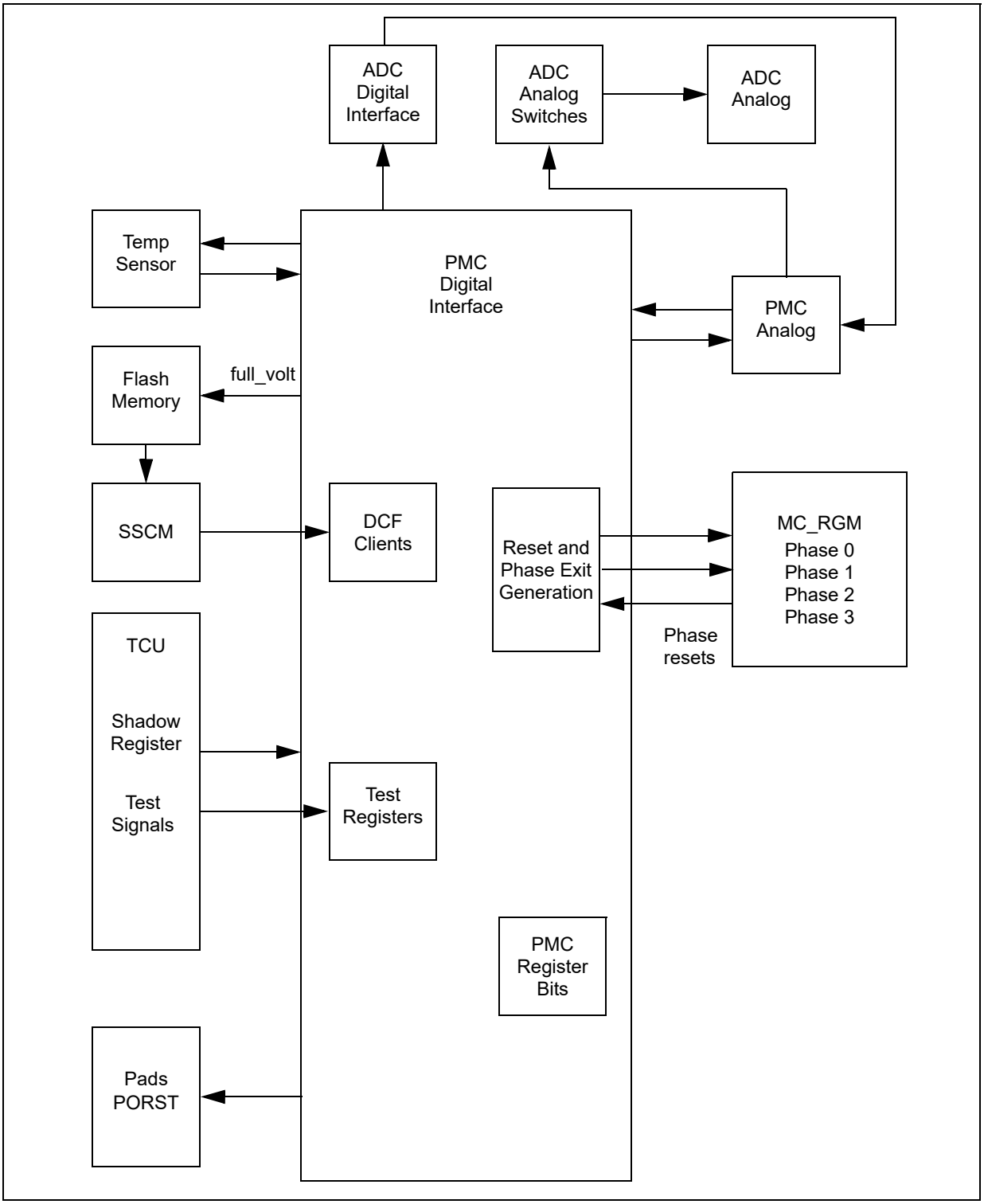
The PMC digital block generates a transfer error event to the system if the selected address exceeds the highest address in the PMC digital memory map, but not if an unused address within the PMC digital memory space is accessed.

Note: The power domains measured by an LVD should not be left unpowered. They must always be connected and in a valid operating range before the device can exit Power-on reset.

54.2 Block diagram

The block diagram for the PMC is shown in *Figure 950*.

Figure 950. Digital PMC block diagram



54.3 Memory map and register definition

Table 914 shows the PMC memory map. The PMC includes many registers for configuring and monitoring LVD monitors and some other control registers.

Table 914. Power Management Controller memory map

Address	Register	Access	Reset Value	Section /Page
PMC_BASE + 0x0000	Supply Gauge Status register (GR_S)	R	0x0000_0000	*54.3.1.1/1537*
PMC_BASE + 0x0004	Pending Gauge Status register (GR_P)	R	0x0000_0000	*54.3.1.2/1538*
PMC_BASE + 0x0008	Interrupt Enable Pending register (IE_P)	R/W	0x0000_0000	*54.3.1.3/1540*
PMC_BASE + 0x0020	LVD098 Event Pending register VD2 (EPR_VD2)	R/W	0x0000_0000	*54.3.1.4/1541*
PMC_BASE + 0x0030	LVD108 Event Pending register VD3 (EPR_VD3)	R/W	0x0000_0000	*54.3.1.5/1542*
PMC_BASE + 0x0034	Reset Event Enable VD3 (REE_VD3)	R/W	0x0000_0008	*54.3.1.6/1543*
PMC_BASE + 0x0038	Reset Event Selection VD3 (RES_VD3)	R/W	0x0000_0000	*54.3.1.7/1543*
PMC_BASE + 0x0090	LVD280 Event Pending register VD9 (EPR_VD9)	R/W	0x0000_0000	*54.3.1.8/1544*
PMC_BASE + 0x00A0	LVD290 Event Pending register VD10 (EPR_VD10)	R/W	0x0000_0000	*54.3.1.9/1544*
PMC_BASE + 0x00A4	Reset Event Enable VD10 (REE_VD10)	R/W	0x0000_C800	*54.3.1.10/1546*
PMC_BASE + 0x00A8	Reset Event Selection VD10 (RES_VD10)	R/W	0x0000_0000	*54.3.1.11/1546*
PMC_BASE + 0x00E0	LVD400 Event Pending register VD14 (EPR_VD14)	R/W	0x0000_0000	*54.3.1.12/1547*
PMC_BASE + 0x00E4	Reset Event Enable VD14 (REE_VD14)	R/W	0x0000_0000	*54.3.1.13/1548*
PMC_BASE + 0x00E8	Reset Event Selection VD14 (RES_VD14)	R/W	0x0000_0000	*54.3.1.14/1549*
PMC_BASE + 0x0104	Voltage Selection for I/O Segment (VSIO)	R/W	0x0000_0000	*54.3.1.15/1550*
PMC_BASE + 0x0240	Main Regulator Control register (MREG_CTRL)	R/W	0x0000_0000	*54.3.1.16/1551*
PMC_BASE + 0x0300	Temperature Event Pending register (EPR_TD)	R/W	0x0000_0000	*54.3.1.17/1551*
PMC_BASE + 0x0304	Temperature Reset Event Enable register (REE_TD)	R/W	0x0000_0000	*54.3.1.18/1552*

Table 914. Power Management Controller memory map(Continued)

Address	Register	Access	Reset Value	Section /Page
PMC_BASE + 0x0308	Temperature Reset Event Selection register (RES_TD)	R/W	0x0000_0000	*54.3.1.19/1553*
PMC_BASE + 0x030C	Temperature detector configuration register (CTL_TD)	R/W	0x0000_0003	*54.3.1.20/1554*

54.3.1 Register descriptions

54.3.1.1 Supply Gauge Status register (GR_S)

This status register contains the gauge of the indicated supply or temperature sensor value, indicating the present state of the voltage or temperature detect events for each of the supplies. Multiple voltage detect signals for a given voltage are combined to form a single bit.

Offset: PMC_BASE + 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	VD14	0	0	0	VD10	VD9	0	0	0	0	0	VD3	0	0	VD2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 951. Supply Gauge register (GR_S)

Table 915. GR_S field descriptions

Field	Description
0–16	Reserved.
17 VD14	VD14 low-voltage detect flag—high voltage 4.00 V supplies. Read-only low-voltage status flag asserted when any of the supplies fall below the LVD threshold and cleared when all the supplies rise above the corresponding LVD threshold. All the 4.00 V supplies are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected.
18–20	Reserved.
21 VD10	VD10 low-voltage detect flag—high voltage 3.00 V supplies. Read-only low-voltage status flag asserted when the supply falls below its LVD threshold and cleared when the supply rises above its LVD threshold. 0 Currently no occurrence. 1 LVD occurrence detected.

Table 915. GR_S field descriptions(Continued)

Field	Description
22 VD9	VD9 low-voltage detect flag—high voltage 2.8 V supplies. Read-only bit is the low-voltage status flag asserted when any of the supplies fall below the LVD threshold and cleared when all the supplies rise above the LVD threshold. All of the 2.8 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected.
23–27	Reserved.
28 VD3	VD3 low-voltage detect flag—low voltage 1.08 V core logic supply. Read-only low-voltage status flag asserted when any of the VD3 supplies fall below the corresponding LVD threshold and cleared when the supplies rise above the corresponding LVD threshold. All the 1.08 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected.
29–30	Reserved.
31 VD2	VD2 low-voltage detect flag—low voltage 0.98 V Core supply Read-only low-voltage status flag asserted when any of the VD2 supplies fall below the LVD threshold and cleared when the supplies rise above the LVD threshold. 0 Currently no occurrence. 1 MVD occurrence detected.

54.3.1.2 Pending Gauge Status register (GR_P)

This configuration register reflects the Event Pending register in a format that includes one bit for each voltage detect signal. These read-only bits indicate whether a voltage level event has occurred at any time in the past. They are cleared by writing ‘1’ in 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 are set.

The raw analog block voltage detect signals are also synchronized to the bus clock before being used for the pending registers.

Offset: PMC_BASE + 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	VD14_A	VD14_IM	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VD10_F	VD10_C	0	VD10_IF	VD10_IJ	0	0	0	VD9_C	0	0	0	VD3_C	0	0	VD2_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 952. Pending Gauge Status register (*GR_P*)

Table 916. GR_P field descriptions

Field	Description
0–8	Reserved.
9 VD14_A	VD14_A low-voltage detect flag—high voltage 4 V ADC supply. 0 No occurrence has happened, or EPR_VD14[LVD14_A] has been cleared. 1 LVD occurrence detected.
10 VD14_IM	VD14_IM low-voltage detect flag—high voltage 4 V I/O Main supply. 0 No occurrence has happened, or EPR_VD14[LVD14_IM] has been cleared. 1 LVD occurrence detected.
11–15	Reserved.
16 VD10_F	VD10_F low-voltage detect flag—high voltage 3 V Flash supply. 0 No occurrence has happened, or EPR_VD10[LVD10_F] has been cleared. 1 LVD occurrence detected.
17 VD10_C	VD10_C low-voltage detect flag—high voltage 2.9 V Core supply. 0 No occurrence has happened, or EPR_VD10[LVD10_C] has been cleared. 1 LVD occurrence detected.
18	Reserved.
19 VD10_IF	VD10_IF low-voltage detect flag—high voltage 2.9 V I/O Ethernet supply. 0 No occurrence has happened, or EPR_VD10[LVD10_IF] has been cleared. 1 LVD occurrence detected.
20 VD10_IJ	VD10_IJ low-voltage detect flag—high voltage 2.9 V I/O JTAG supply. 0 No occurrence has happened, or EPR_VD10[LVD10_IJ] has been cleared. 1 LVD occurrence detected.
21–23	Reserved.
24 VD9_C	VD9_C low-voltage detect flag—high voltage 2.7 V cold point supply. 0 No occurrence has happened, or EPR_VD9[LVD9_C has been cleared. 1 LVD occurrence detected.
25–27	Reserved.

Table 916. GR_P field descriptions(Continued)

Field	Description
28 VD3_C	VD3_C low-voltage detect flag—low voltage Core cold point 1.08 V supply. 0 No occurrence has happened, or EPR_VD3[LVD3_C] has been cleared. 1 LVD occurrence detected.
29–30	Reserved.
31 VD2_C	VD2_C low-voltage detect flag—low voltage 0.98 V Core hot point supply. 0 No occurrence has happened, or EPR_VD2[LVD2_C] has been cleared. 1 LVD occurrence detected.

54.3.1.3 Interrupt Enable Pending register (IE_P)

This configuration register contains a set of Interrupt Enable bits that correspond to each of the Event Pending registers in a similar format to the Pending Gauge Status register (GR_P). These bits toggle whether an interrupt occurs when the voltage event is seen:

Writing ‘1’ to the MSB (IE_P[IE_EN]) enables all of the voltage detect bits in the register to be read or written at any time.

Offset: PMC_BASE + 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IE_EN	0	0	0	0	0	0	0	0	VD14IE_A	VD14IE_IM	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VD10IE_F	VD10IE_C	0	VD10IE_IF	VD10IE_IJ	0	0	0	VD9IE_C	0	0	0	VD3IE_C	0	0	VD2IE_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 953. Interrupt Enable Pending register (*IE_P*)

Table 917. IE_P field descriptions

Field	Description
0 IE_EN	IE_EN. Interrupt Enable. 0 No interrupt enables can be written. 1 Any interrupt enable can be written.
1–8	Reserved.
9 VD14IE_A	VD14IE_A Interrupt Enable. 0 No interrupt occurs. 1 An interrupt occurs on a voltage detect event.
10 VD14IE_IM	VD14IE_IM Interrupt Enable. 0 No interrupt occurs. 1 An interrupt occurs on a voltage detect event.

Table 917. IE_P field descriptions(Continued)

Field	Description
11–15	Reserved.
16 VD10IE_F	VD10IE_F Interrupt Enable. 0 No interrupt occurs. 1 An interrupt occurs on a voltage detect event.
17 VD10IE_C	VD10IE_C Interrupt Enable. 0 No interrupt occurs. 1 An interrupt occurs on a voltage detect event.
18	Reserved.
19 VD10IE_IF	VD10IE_IF Interrupt Enable. 0 No interrupt occurs. 1 An interrupt occurs on a voltage detect event.
20 VD10IE_IJ	VD10IE_IJ Interrupt Enable. 0 No interrupt occurs. 1 An interrupt occurs on a voltage detect event.
21–23	Reserved.
24 VD9IE_C	VD9IE_C Interrupt Enable. 0 No interrupt occurs. 1 An interrupt occurs on a voltage detect event.
25–27	Reserved.
28 VD3IE_C	VD3IE_C Interrupt Enable. 0 No interrupt occurs. 1An interrupt occurs on a voltage detect event.
29–30	Reserved.
31 VD2IE_C	VD2IE_C Interrupt Enable. 0 No interrupt occurs. 1 An interrupt occurs on a voltage detect event.

54.3.1.4 Event Pending register (EPR_VD2)

This register indicates the present and past state of the voltage detect signals. If the voltage detect event crosses the trigger threshold after the last clearing event, the flag bit is set. Write ‘1’ to clear the flag. Since this voltage event can only be seen when the LVD096 signals assert, these flags can only be seen in test modes that disable these PORs.

Offset: PMC_BASE + 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	0	0	0	0	0	LVD2_C
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 954. Event Pending register (*EPR_VD2*)

Table 918. EPR_VD2 field descriptions

Field	Description
0–30	Reserved.
31 LVD2_C	LVD2_C low-voltage status flag—low voltage 0.98 V Core cold point supply. 0 Currently no occurrence, or previously cleared by writing ‘1’. 1 LVD occurrence detected.

54.3.1.5 Event Pending register (EPR_VD3)

This register indicates the present and past state of the voltage detect signals. If the voltage detect event crosses the trigger threshold after the last clearing event, the flag bit is set asynchronously. Write ‘1’ to clear the flag bit.

Offset: PMC_BASE + 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	0	0	LVD3_C	0	0	0
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 955. Event Pending register (*EPR_VD3*)

Table 919. EPR_VD3 field descriptions

Field	Description
0–27	Reserved.
28 LVD3_C	LVD3_C low-voltage status flag—low voltage 1.08 V Core hot point supply. IE_P[VD3IE_C] *enables* a low-voltage interrupt. REE_VD3[LVD3_C] *enables* a system reset (which clears and negates the interrupt). RES_VD3[LVD3_C] *selects* functional or destructive reset. 0 Currently no occurrence, or previously cleared by writing '1'. 1LVD occurrence detected.
29–31	Reserved.

54.3.1.6 Reset Event Enable register (REE_VD3)

This register controls whether the voltage detect signal event causes a reset. Setting this bit causes a reset when the selected voltage passes the trigger event. These bits are loaded from the Flash during boot and cannot be disabled (cleared) if enabled (set) when loaded.

Offset: PMC_BASE + 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	0	0	0	0	0	0	0	0	0	0	0	0	LVD3_C	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 956. Reset Event Enable register (*REE_VD3*)

Table 920. REE_VD3 field descriptions

Field	Description
0–27	Reserved.
28 LVD3_C	LVD3_C reset enable—LVD assertion on low voltage core 1.08 V hot point supply RES_VD3[LVD3_C] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
29–31	Reserved.

54.3.1.7 Reset Event Select register (RES_VD3)

This register controls whether the voltage detect signal event causes a destructive or functional reset. These bits are loaded from the Flash during the boot sequence.

Offset: PMC_BASE + 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	0	0	LVD3_C	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 957. Reset Event Select register (*RES_VD3*)

Table 921. RES_VD3 field descriptions

Field	Description
0–27	Reserved.
28 LVD3_C	LVD3_C reset event select—LVD on the low voltage Core 1.08 V hot point supply. The REE_VD3[LVD3_C] *enables* a system reset. 0 Destructive reset. 1 Functional Reset.
29–31	Reserved.

54.3.1.8 Event Pending register (EPR_VD9)

This Event Pending register indicates the present and past state of the voltage detect signals. If the voltage detect event crosses the trigger threshold after the last clearing event, the flag bit is set asynchronously. Write ‘1’ to clear the flag.

54.3.1.9 Event Pending register (EPR_VD10)

This Event Pending register indicates the present and past state of the voltage detect signals. If the voltage detect event crosses the trigger threshold after the last clearing event, the flag bit is set asynchronously. Write ‘1’ to clear the flag.

Offset: PMC_BASE + 0x00A0 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	LVD10_F	LVD10_C	0	LVD10_IF	LVD10_IJ	0	0	0	0	0	0	0	0	0	0	0
W	w1c	w1c		w1c	w1c											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 958. Event Pending register (*EPR_VD10*)

Table 922. EPR_VD10 field descriptions

Field	Description
0–15	Reserved.
16 LVD10_F	LVD10_F low-voltage status flag—high voltage 2.90V Flash supply. IE_P[VD10IE_F] *enables* a low-voltage interrupt. REE_VD10[LVD10_F] *enables* a system reset (which clears and negates the interrupt). RES_VD10[LVD10_F] *selects* functional or destructive reset. 0 Currently no occurrence, or previously cleared by writing ‘1’. 1LVD occurrence detected on the high voltage flash supply.
17 LVD10_C	LVD10_C low-voltage status flag—high voltage 2.90V CPU supply. IE_P[VD10IE_C] *enables* a low-voltage interrupt. REE_VD10[LVD10_C] *enables* a system reset (which clears and negates the interrupt). RES_VD10[LVD10_C] *selects* functional or destructive reset. 0 Currently no occurrence, or previously cleared by writing ‘1’. 1 LVD occurrence detected on the high voltage flash supply.
18	Reserved.
19 LVD10_IF	LVD10_IF low-voltage status flag—high voltage 2.90V Ethernet supply. IE_P[VD10IE_IF] *enables* a low-voltage interrupt. REE_VD10[LVD10_IF] *enables* a system reset (which clears and negates the interrupt). RES_VD10[LVD10_IF] *selects* functional or destructive reset. 0 Currently no occurrence, or previously cleared by writing ‘1’. 1 LVD occurrence detected on the high voltage flash supply.
20 LVD10_IJ	LVD10_IJ low-voltage status flag—high voltage 2.90V IO/JTAG supply. IE_P[VD10IE_IJ] *enables* a low-voltage interrupt. REE_VD10[LVD10_IJ] *enables* a system reset (which clears and negates the interrupt). RES_VD10[LVD10_IJ] *selects* functional or destructive reset. 0 Currently no occurrence, or previously cleared by writing ‘1’. 1 LVD occurrence detected on the high voltage flash supply.
21–31	Reserved.

54.3.1.10 Reset Event Enable register (REE_VD10)

This Reset Event Enable register controls whether the voltage detect signal event causes a reset. These bits are loaded from the Flash during boot and cannot be disabled (cleared) if enabled (set) when loaded.

Offset: PMC_BASE + 0x00A4 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	LVD10_F	LVD10_C	0	LVD10_IF	LVD10_IJ	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 959. Reset Event Enable register (*REE_VD10*)

Table 923. REE_VD10 field descriptions

Field	Description
0–15	Reserved.
16 LVD10_F	LVD10_F reset enable—LVD on the Flash supply. RES_VD10[LVD10_F] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
17 LVD10_C	LVD10_C reset enable—LVD on the core supply RES_VD10[LVD10_C] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
18	Reserved.
19 LVD10_IF	LVD10_IF reset disable—LVD on the Ethernet supply. RES_VD10[LVD10_IF] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
20 LVD10_IJ	LVD10_IJ reset enable—LVD on the JTAG supply. RES_VD10[LVD10_IJ] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
21–31	Reserved.

54.3.1.11 Reset Event Select register (RES_VD10)

This Reset Event Select register controls whether the voltage detect signal event causes a destructive or functional reset. Setting this bit 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: PMC_BASE + 0x00A8 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	LVD10_F	LVD10_C	0	LVD10_IF	LVD10_IJ	0	0	0	0	0	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 960. Reset Event Select register (RES_VD10)

Table 924. RES_VD10 field descriptions

Field	Description
0–15	Reserved.
16 LVD10_F	LVD10_F reset enable—LVD on the Flash supply. RES_VD10[LVD10_F] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
17 LVD10_C	LVD10_C reset enable—LVD on the Core supply. RES_VD10[LVD10_C] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
18	Reserved.
19 LVD10_IF	LVD10_IF reset disable—LVD on the Ethernet supply. RES_VD10[LVD10_IF] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
20 LVD10_IJ	LVD10_IJ reset enable—LVD on the JTAG supply. RES_VD10[LVD10_IJ] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
21–31	Reserved.

54.3.1.12 Event Pending register (EPR_VD14)

This Event Pending register indicates the present and past state of the voltage detect signals. If the voltage detect event crosses the trigger threshold after the last clearing event, the flag bit is set asynchronously. Write ‘1’ to clear the flag.

Offset: PMC_BASE + 0x00E0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LVD14_A	LVD14_IM	0	0	0	0	0
W										w1c	w1c					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	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 961. Event Pending register (*EPR_VD14*)

Table 925. EPR_VD14 field descriptions

Field	Description
0–8	Reserved.
9 LVD14_A	LVD14_A low-voltage status flag—high voltage 4 V ADC supply. IE_P[VD14IE_A] *enables* a low-voltage interrupt. REE_VD14[LVD14_A] *enables* a system reset (which clears and negates the interrupt). RES_VD14[LVD14_A] *selects* functional or destructive reset. 0 Currently no occurrence, or previously cleared by writing ‘1’. 1 LVD occurrence detected on the high voltage ADC supply.
10 LVD14_IM	LVD14_IM low-voltage status flag—high voltage 4 V I/O Main supply. IE_P[VD14IE_IM] *enables* a low-voltage interrupt. REE_VD14[LVD14_IM] *enables* a system reset (which clears and negates the interrupt). RES_VD14[LVD14_IM] *selects* functional or destructive reset. 0 Currently no occurrence, or previously cleared by writing ‘1’. 1 LVD occurrence detected on the high voltage I/O Main supply.
11–31	Reserved.

54.3.1.13 Reset Event Enable register (REE_VD14)

This Reset Event Enable register controls whether the voltage detect signal event causes a reset. These bits are loaded from the Flash during boot and cannot be disabled (cleared) if enabled (set) when loaded.

Offset: PMC_BASE + 0x00E4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LVD14_A	LVD14_IM	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1(1)	1(1)	0	0	0	0	0

	16	17	18	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 962. Reset Event Enable register (REE_VD14)

1. The value of this field depends on the value programmed in TEST flash. The default value is ‘1’.

Table 926. REE_VD14 field descriptions

Field	Description
0–8	Reserved.
9 LVD14_A	LVD14_A reset enable—LVD on the ADC supply. RES_VD14[LVD14_A] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
10 LVD14_IM	LVD14_IM reset enable—LVD on the I/O Main supply. RES_VD14[LVD14_IM] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
11–31	Reserved.

54.3.1.14 Reset Event Select register (RES_VD14)

This Reset Event Select register controls whether the voltage detect signal event causes a destructive or functional reset. Setting this bit 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: PMC_BASE + 0x00E8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LVD14_A	LVD14_IM	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	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 963. Reset Event Select register (*RES_VD14*)

Table 927. RES_VD14 field descriptions

Field	Description
0–8	Reserved.
9 LVD14_A	LVD14_A reset event select—LVD on the ADC supply. REE_VD14[LVD14_A] *enables* the reset event. 0 Destructive reset 1 Functional reset
10 LVD14_IM	LVD14_IM reset event select—LVD on the I/O Main supply. REE_VD14[LVD14_IM] *enables* reset event. 0 Destructive reset 1 Functional reset.
11–31	Reserved.

54.3.1.15 Voltage Supply for I/O Segments register (VSIO)

The Voltage Supply for I/O Segments (VSIO) register indicates if a particular I/O segment uses a 3.3 V or 5.0 V supply. These bits all reset to use a 5.0 V supply.

Offset: PMC_BASE + 0x0104 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	VSIO_IF	VSIO_IJ	0	0	0	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 964. Voltage Supply for I/O Segments register (*VSIO*)

Table 928. VSIO field descriptions

Field	Description
0–20	Reserved.
21 VSIO_IF	VSIO_IF control—Ethernet I/O segment supply. 0 The I/O segment uses a 3.3 V supply. 1 The I/O segment uses a 5.0 V supply.
22 VSIO_IJ	VSIO_IJ control—JTAG I/O segment supply. 0 The I/O segment uses a 3.3 V supply. 1 The I/O segment uses a 5.0 V supply.
23–31	Reserved.

54.3.1.16 Main Regulator Control register (MREG_CTRL)

This Control register controls and indicates the state of the Main Regulator for SPC572Lx. If the MREG_ENB signal is low, the main regulator for SPC572Lx is enabled. If the MREG_ENB signal is high, the main regulator is disabled.

Offset: PMC_BASE + 0x0240 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	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 965. Main Regulator Control register (MREG_CTRL)

Table 929. MREG_CTRL field descriptions

Field	Description
0–30	Reserved.
31 MREG_ENB	MREG_ENB—Main Regulator for SPC572Lx. 0 Main Regulator is enabled. 1 Main Regulator is disabled.

54.3.1.17 Event Pending register (EPR_TD)

This Event Pending register indicates the present and past state of the temperature sensor signals. The flag bit is set when the temperature event passes the trigger threshold after the last clearing event. Write ‘1’ to clear the flag.

Offset: PMC_BASE + 0x0300 Access: User read/write

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

	16	17	18	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
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 966. Event Pending register (*EPR_TD*)

Table 930. EPR_TD field descriptions

Field	Description
0–28	Reserved.
29 TEMP_2	TEMP_2 temperature status flag—high temperature sensor input 2 (165 °C). – CTL_TD[TS2IE] *enables* a low-voltage interrupt. – REE_TD[TEMP_2] *enables* a system reset (which clears and negates the interrupt). – RES_TD[TEMP_2] *selects* functional or destructive reset. 0 Currently no occurrence, or previously cleared by writing ‘1’. 1 Temperature occurrence detected.
30 TEMP_1	TEMP_1 temperature status flag—high temperature sensor input 1 (150 °C). – CTL_TD[TS1IE] *enables* a low-voltage interrupt. – REE_TD[TEMP_1] *enables* a system reset (which clears and negates the interrupt). – RES_TD[TEMP_1] *selects* functional or destructive reset. 0 Currently no occurrence, or previously cleared by writing ‘1’. 1 Temperature occurrence detected.
31 TEMP_0	TEMP_0 temperature status flag—low temperature sensor point 0 (–40 °C). – CTL_TD[TS0IE] *enables* a low-voltage interrupt. – REE_TD[TEMP_0] *enables* a system reset (which clears and negates the interrupt). – RES_TD[TEMP_0] *selects* functional or destructive reset. 0 Currently no occurrence, or previously cleared by writing ‘1’. 1 Temperature occurrence detected.

54.3.1.18 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 when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then the same REE and RES bits are also cleared by the reset.

Offset: PMC_BASE + 0x0304 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_2	TEMP_1	TEMP_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 967. Reset Event Enable register (*REE_TD*)

Table 931. REE_TD field descriptions

Field	Description
0–28	Reserved.
29 TEMP_2	TEMP_2 reset enable—high temperature sensor input 2 (165 °C). RES_TD[TEMP_2] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
30 TEMP_1	TEMP_1 reset enable—high temperature sensor input 1 (150 °C). RES_TD[TEMP_1] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.
31 TEMP_0	TEMP_0 reset enable—low temperature sensor input 0 (–40 °C). RES_TD[TEMP_0] *selects* functional or destructive reset. 0 Disabled. 1 Enabled.

54.3.1.19 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 when the corresponding Reset Event Enable field is set.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then the same REE and RES bits are also cleared by the reset.

Offset: PMC_BASE + 0x0308 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_2	TEMP_1	TEMP_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 968. Reset Event Select register (*RES_TD*)

Table 932. RES_TD field descriptions

Field	Description
0–28	Reserved.
29 TEMP_2	TEMP_2 reset event select—high temperature sensor input 2 (165 °C). REE_TD[TEMP_2] *enables* the reset event. 0 Destructive reset. 1 Functional reset.
30 TEMP_1	TEMP_1 reset event select—high temperature sensor input 1 (150 °C). REE_TD[TEMP_1] *enables* the reset event. 0 Destructive reset. 1 Functional reset.
31 TEMP_0	TEMP_0 reset event select—low temperature sensor 0 (–40 °C). REE_TD[TEMP_0] *enables* the reset event. 0 Destructive reset. 1 Functional reset.

54.3.1.20 Temperature Sensor Configuration register (CTL_TD)

The Temperature Sensor register (CTL_TD) contains two enables that must both be set in order for the Temperature Sensor to be operable. It also includes two 4-bit signed trim adjustment register fields that are used to modify both Temperature Sensor trim values (one bus for over-temperature and one bus for under-temperature). Finally, this register includes Interrupt Enables (IE) for each of the Temperature Sensor EPR bits.

TRIM_ADJ_OVER and TRIM_ADJ_UNDER are used to introduce additional hysteresis into the Temp Sensor events.

For instance, the TRIM_ADJ_OVER bits could be written to 11111 (-7) and the Temperature Sensor event set as an interrupt. When this interrupt triggers, the TRIM_ADJ_OVER bits could be updated to 0111 (+7), the interrupt cleared, and the Temperature Sensor event set as a reset.

In this way, a *warning* interrupt could be established before a *shutdown* event (at a higher temperature).

Care must be taken to include a timeout during the clearing of the interrupt described in the example above. The Temperature Sensor has hysteresis and a timeout may be necessary to indicate if the interrupt flag has not cleared in a reasonable amount of time. If this is the case, then an error should be generated.

Offset: PMC_BASE + 0x030C 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	TRIM_ADJ_OVER				0	0	TRIM_ADJ_UNDER				DOUT_EN	AOUT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 969. Temperature Sensor Configuration register (CTL_TD)

Table 933. CTL_TD field descriptions

Field	Description
0–15	Reserved.
16 TS2IE	TS2IE Interrupt Enable—Temperature Sensor input 2. The MSB in IE_P must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.
17 TS1IE	TS1IE Interrupt Enable—Temperature Sensor input 1. The MSB in IE_P must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.
18 TS0IE	TS0IE Interrupt Enable—Temperature Sensor input 0. The MSB in IE_P must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.
19	Reserved.
20–23 TRIM_ADJ_OVER	TRIM_ADJ_OVER[3:0]—Customer adjustable over trim register. These bits are a signed binary number that is added to the over temperature trim (NT_TD2) value. The TRIM value is protected from overflows and underflows due to this addition.
24–25	Reserved.
26–29 TRIM_ADJ_UNDER	TRIM_ADJ_UNDER[3:0]—Customer adjustable under trim register. These bits are a signed binary number that is added to the under temperature trim value (NT_TD0). The TRIM value is protected from overflows and underflows due to this addition.

Table 933. CTL_TD field descriptions(Continued)

Field	Description
30 DOUT_EN	DOUT_EN. Digital Output Enable.
31 AOUT_EN	AOUT_EN. Analog Output Enable.

54.4 Analog PMC interface

The outputs from the PMC digital to the PMC analog circuit consist of trim registers for each LVD circuit, an enable for each LVD (tied to the enabled state), and ADC channel select controls. The ADC channel select controls are described in more detail in the ADC interface section of this chapter, and in the Successive Approximation register Analog-to-Digital Converter (SARADC) Digital Interface chapter.

The signals from the PMC analog to the PMC digital logic are the POR and LVD event indication signals (both active high and active low, but only active low are used). These signals are used by the PMC digital logic to create the appropriate resets for the rest of the system.

There is a settling time for the analog voltage detect circuits specified at 5 microseconds. The PMC digital counts clock edges after the loading of new flash trim values to verify that this time has passed. This time can vary from 5 microseconds to 11.4 microseconds.

54.5 Temperature Sensor Interface Logic

The Temperature Sensor interface consists of two enables: an analog and digital enable, a set of mode registers, two 7-bit trim registers (sometimes referenced as one 14-bit trim register), and an overtemp threshold select control.

The Temperature Sensor outputs three signals to the PMC digital logic: over_temp1, over_temp2, and under_temp signals. These signals are used by the PMC digital logic to create the appropriate resets for the rest of the system.

There are two 4-bit signed registers (TRIM_ADJ_OVER[3:0] and TRIM_ADJ_UNDER[3:0]) that software uses to adjust the value of the Temp Sensor Trim values. The fourth bit of these registers indicates the sign.

TRIM_ADJ_OVER is added to TS2_TRIM and TRIM_ADJ_UNDER is added to TS0_TRIM. This addition is protected from overflow and underflow (the TRIM value remains at max or min value).

The test interface is discussed in the test section.

For more information regarding the Temperature Sensor function, refer to the Temperature Sensor chapter.

54.6 Power On Reset (MC_RGM Phase Gates)

The PMC digital logic interfaces to the Reset Generation Module in order to indicate the type and conditions of the resets from the PMC and Temperature Sensor Analog blocks.

There are two signals for each the PMC and Temp Sensor blocks: one is used to indicate a destructive reset, and one is used to indicate a functional reset.

The PMC digital logic also indicates when the RGM may allow transition to the next POR phase.

The digital PMC 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 PMC digital logic uses all the enabled Voltage Detects to determine whether it may exit Phase 3. Once all the enabled LVDs have become deasserted and the time for the LVD circuits to adjust to new trim values has elapsed, the signal is sent to the RGM that the necessary voltage levels to exit Phase 3 have been achieved.

There are five POR signals: POR_VDDREG, POR041b_c, POR200b_c, POR098C, and POR270C. All of these except POR_VDDREG and POR041b can be disabled in test mode. During Test Mode, there are pending flags (EPR) that can be seen. These correspond to the POR098C (VD2C), and POR270C.

SPC572Lx includes two new POR signals MVD085TB_C and MVD240TB_C.

54.7 Flash interface (DCF client usage)

The PMC digital is updated during POR from the Flash block via a DCF client. This update operation loads several pieces of data, most of which being trim values applied to analog modules (PMC, Temperature Sensor, and ADC Bandgap).

The 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.

The updated values for the REE and RES bits are also loaded during the DCF single read process. See *Section 54.3.1.18: 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 (allowing the bits to be rewriteable), but the reset value of the REE register bits are all high (enabling resets out of POR).

The loaded REE DCF bits 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, refer to the DCF chapter.

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 PMC digital via the SSCM module.

Once the PMC 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, refer to the Flash chapter.

55 Power Control Unit (MC_PCU)

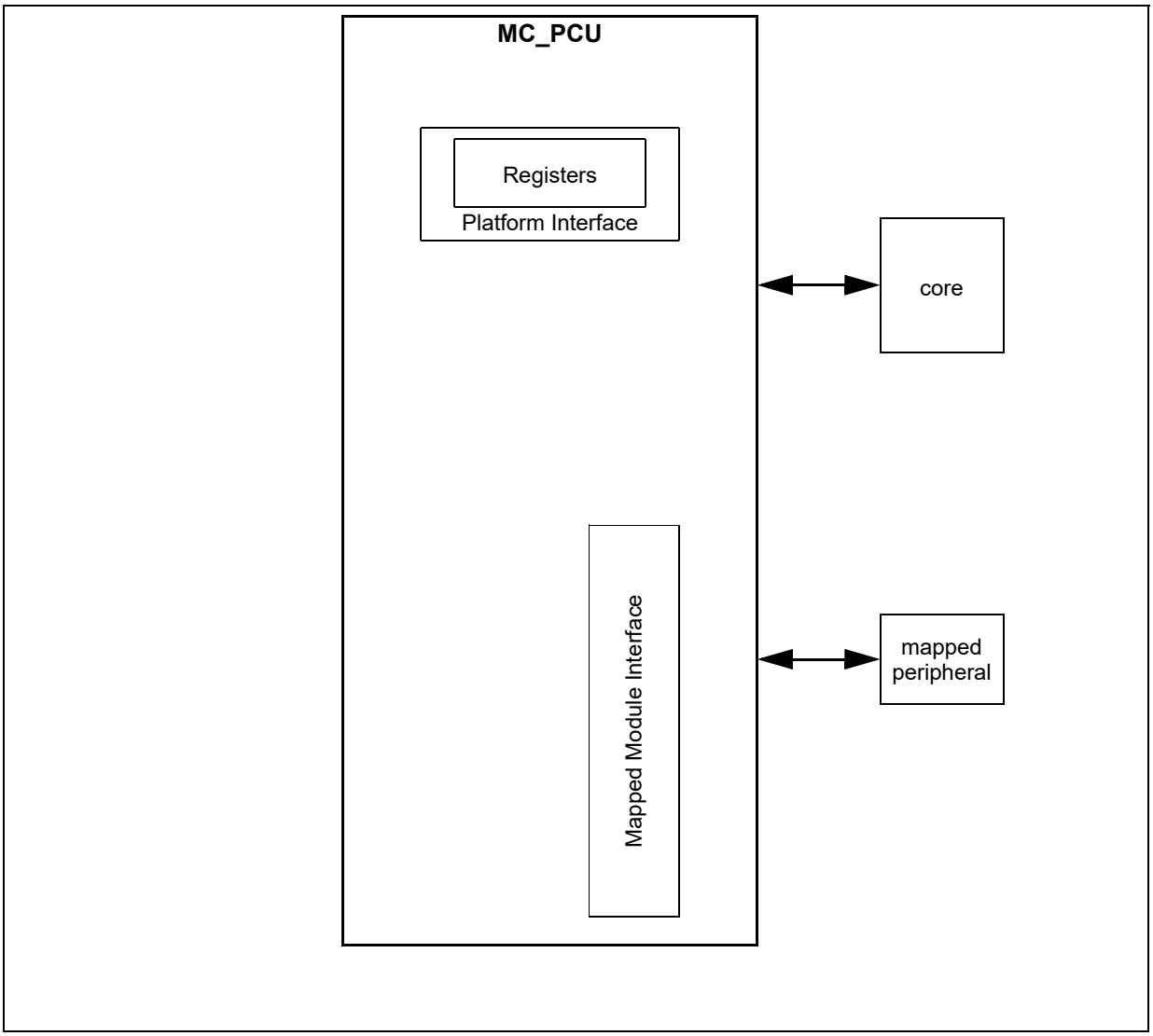
55.1 Introduction

55.1.1 Overview

The power control unit (MC_PCU) acts as a bridge for mapping the PMC peripheral to the MC_PCU address space.

Figure 970 shows the MC_PCU block diagram.

Figure 970. MC_PCU block diagram



55.1.2 Features

The MC_PCU includes the following feature:

- maps the PMC registers to the MC_PCU address space

55.2 External signal description

The MC_PCU has no connections to any external pins.

55.3 Memory map and register definition

55.3.1 Memory map

Table 934. MC_PCU register description

Address	Register	Size	Access			Location
			User	Supervisor	Test	
0x0040	Power Domain Status Register (PCU_PSTAT)	word	read	read	read	*on page 1560*

Note: Any access to unused registers as well as write accesses to read-only registers will:

- *not change register content*
- *cause a transfer error*

55.3.2 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.

55.3.2.1 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 971. Power Domain Status Register (PCU_PSTAT)

This register reflects the power status of all available power domains.

Table 935. Power Domain Status Register (PCU_PSTAT) field descriptions

Field	Description
PD*0*	Power status for power domain *0* 0Power domain is inoperable 1Power domain is operable

56 Mode Entry Module (MC_ME)

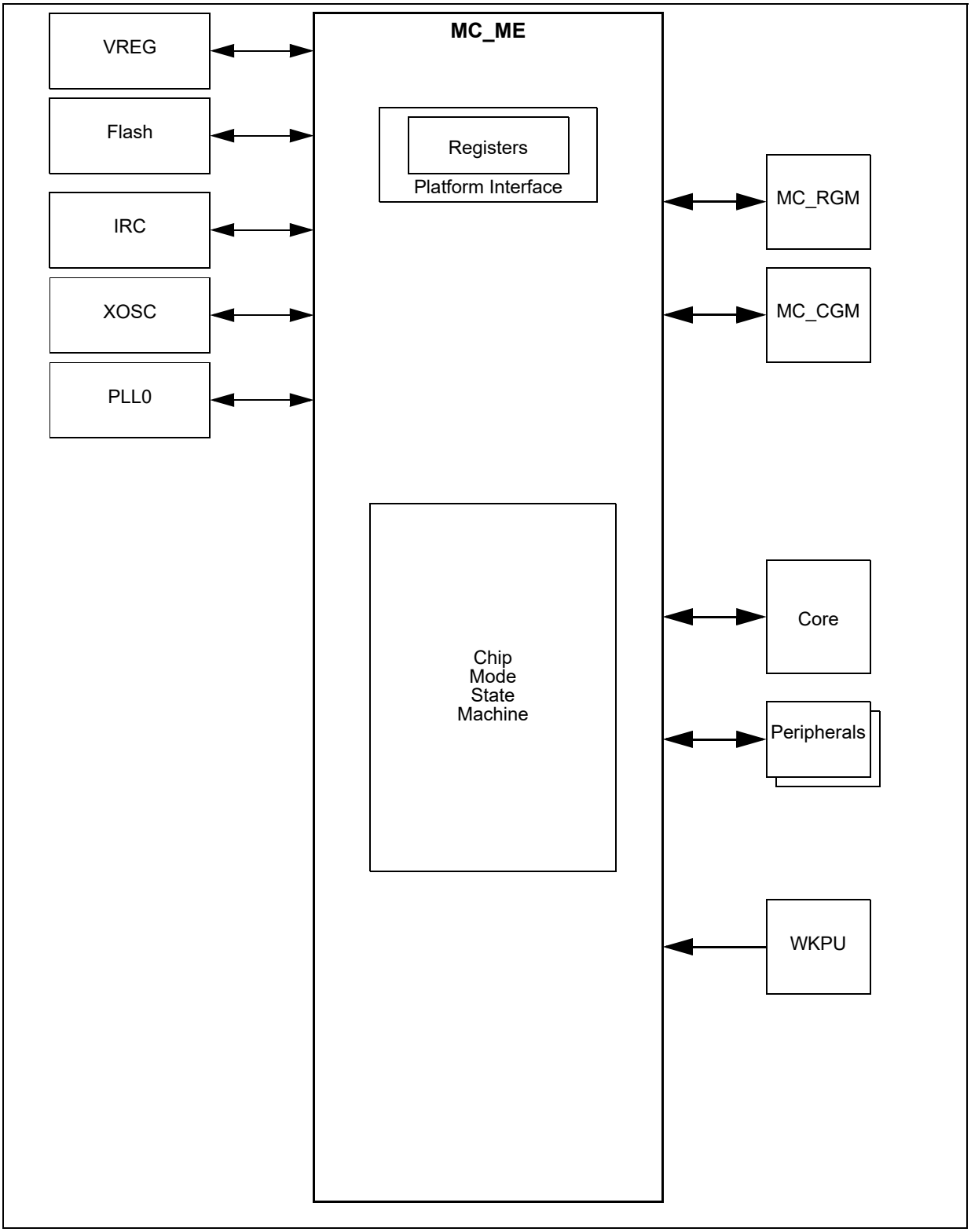
56.1 Introduction

56.1.1 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 972 shows the MC_ME block diagram.

Figure 972. MC_ME block diagram



56.1.2 Features

The MC_ME includes the following features:

- control of the available modes by the **ME_ME** register
- definition of various chip mode configurations by the **ME_<*mode*>_MC** registers
- control of the actual chip mode by the **ME_MCTL** register
- capture of the current mode and various resource status within the contents of the **ME_GS** register
- optional generation of various mode transition interrupts
- status bits for each cause of invalid mode transitions
- peripheral clock gating control based on the **ME_RUN_PC0…7**, **ME_LP_PC0…7**, and **ME_PCTL*n*** registers
- additional core clock gating and boot address control based on the **ME_CCTL*n*** and **ME_CADDR*n*** registers
- capture of current peripheral and additional core clock gated/enabled status
- progressive system clock switching when transitioning from a lower power consumption mode to a higher power consumption mode, and vice versa

56.1.3 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 936 describes the MC_ME modes.

Table 936. 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. BAM when present is executed in DRUN mode.	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 936. 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 pre-defined 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 & 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

56.2 External signal description

The MC_ME has no connections to any external pins.

56.3 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

56.3.1 Memory map

Table 937. MC_ME register description

Address	Name	Size	Access			Location
			User	Supervisor	Test	
0x000	Global Status Register (ME_GS)	word	read	read	read	on page 1569
0x004	Mode Control Register (ME_MCTL)	word	read	read/write	read/write	on page 1571
0x008	Mode Enable Register (ME_ME)	word	read	read/write	read/write	on page 1572
0x00C	Interrupt Status Register (ME_IS)	word	read	read/write	read/write	on page 1574
0x010	Interrupt Mask Register (ME_IM)	word	read	read/write	read/write	on page 1575
0x014	Invalid Mode Transition Status Register (ME_IMTS)	word	read	read/write	read/write	on page 1576
0x018	Debug Mode Transition Status Register (ME_DMTS)	word	read	read	read	on page 1577
0x020	RESET Mode Configuration Register (ME_RESET_MC)	word	read	read	read	on page 1581
0x024	TEST Mode Configuration Register (ME_TEST_MC)	word	read	read/write	read/write	on page 1581
0x028	SAFE Mode Configuration Register (ME_SAFE_MC)	word	read	read/write	read/write	on page 1582
0x02C	DRUN Mode Configuration Register (ME_DRUN_MC)	word	read	read/write	read/write	on page 1582
0x030	RUN0 Mode Configuration Register (ME_RUN0_MC)	word	read	read/write	read/write	on page 1583
0x034	RUN1 Mode Configuration Register (ME_RUN1_MC)	word	read	read/write	read/write	on page 1583
0x038	RUN2 Mode Configuration Register (ME_RUN2_MC)	word	read	read/write	read/write	on page 1583
0x03C	RUN3 Mode Configuration Register (ME_RUN3_MC)	word	read	read/write	read/write	on page 1583
0x040	HALT0 Mode Configuration Register (ME_HALT0_MC)	word	read	read/write	read/write	on page 1583
0x048	STOP0 Mode Configuration Register (ME_STOP0_MC)	word	read	read/write	read/write	on page 1584
0x060	Peripheral Status 0 Register (ME_PS0)	word	read	read	read	on page 1586
0x064	Peripheral Status 1 Register (ME_PS1)	word	read	read	read	on page 1586

Table 937. MC_ME register description(Continued)

Address	Name	Size	Access			Location
			User	Supervisor	Test	
0x068	Peripheral Status 2 Register (ME_PS2)	word	read	read	read	on page 1587
0x06C	Peripheral Status 3 Register (ME_PS3)	word	read	read	read	on page 1587
0x070	Peripheral Status 4 Register (ME_PS4)	word	read	read	read	on page 1588
0x080	Run Peripheral Configuration 0 Register (ME_RUN_PC0)	word	read	read/write	read/write	on page 1588
0x084	Run Peripheral Configuration 1 Register (ME_RUN_PC1)	word	read	read/write	read/write	on page 1588
0x088	Run Peripheral Configuration 2 Register (ME_RUN_PC2)	word	read	read/write	read/write	on page 1588
0x08C	Run Peripheral Configuration 3 Register (ME_RUN_PC3)	word	read	read/write	read/write	on page 1588
0x090	Run Peripheral Configuration 4 Register (ME_RUN_PC4)	word	read	read/write	read/write	on page 1588
0x094	Run Peripheral Configuration 5 Register (ME_RUN_PC5)	word	read	read/write	read/write	on page 1588
0x098	Run Peripheral Configuration 6 Register (ME_RUN_PC6)	word	read	read/write	read/write	on page 1588
0x09C	Run Peripheral Configuration 7 Register (ME_RUN_PC7)	word	read	read/write	read/write	on page 1588
0x0A0	Low-Power Peripheral Configuration 0 Register (ME_LP_PC0)	word	read	read/write	read/write	on page 1589
0x0A4	Low-Power Peripheral Configuration 1 Register (ME_LP_PC1)	word	read	read/write	read/write	on page 1589
0x0A8	Low-Power Peripheral Configuration 2 Register (ME_LP_PC2)	word	read	read/write	read/write	on page 1589
0x0AC	Low-Power Peripheral Configuration 3 Register (ME_LP_PC3)	word	read	read/write	read/write	on page 1589
0x0B0	Low-Power Peripheral Configuration 4 Register (ME_LP_PC4)	word	read	read/write	read/write	on page 1589
0x0B4	Low-Power Peripheral Configuration 5 Register (ME_LP_PC5)	word	read	read/write	read/write	on page 1589
0x0B8	Low-Power Peripheral Configuration 6 Register (ME_LP_PC6)	word	read	read/write	read/write	on page 1589
0x0BC	Low-Power Peripheral Configuration 7 Register (ME_LP_PC7)	word	read	read/write	read/write	on page 1589
0x0C9	LFAST_0 Control Register (ME_PCTL9)	byte	read	read/write	read/write	on page 1590

Table 937. MC_ME register description(Continued)

Address	Name	Size	Access			Location
			User	Supervisor	Test	
0x0CB	SIPI_0 Control Register (ME_PCTL11)	byte	read	read/write	read/write	on page 1590
0x0DD	DECIFILTER Control Register (ME_PCTL29)	byte	read	read/write	read/write	on page 1590
0x0DE	PIT_RTI_0 Control Register (ME_PCTL30)	byte	read	read/write	read/write	on page 1590
0x0DF	PIT_RTI_1 Control Register (ME_PCTL31)	byte	read	read/write	read/write	on page 1590
0x0E4	DMAMUX_0 Control Register (ME_PCTL36)	byte	read	read/write	read/write	on page 1590
0x0FB	ADCSD_2 Control Register (ME_PCTL59)	byte	read	read/write	read/write	on page 1590
0x0FE	CCCU Control Register (ME_PCTL62)	byte	read	read/write	read/write	on page 1590
0x105	MCAN_2 Control Register (ME_PCTL69)	byte	read	read/write	read/write	on page 1590
0x106	MCAN_1 Control Register (ME_PCTL70)	byte	read	read/write	read/write	on page 1590
0x10A	CAN_RAM_CTRL Control Register (ME_PCTL74)	byte	read	read/write	read/write	on page 1590
0x115	LINFlexD_14 Control Register (ME_PCTL85)	byte	read	read/write	read/write	on page 1590
0x11B	LINFlexD_1 Control Register (ME_PCTL91)	byte	read	read/write	read/write	on page 1590
0x11C	LINFlexD_0 Control Register (ME_PCTL92)	byte	read	read/write	read/write	on page 1590
0x121	DSPI_4 Control Register (ME_PCTL97)	byte	read	read/write	read/write	on page 1590
0x123	DSPI_0 Control Register (ME_PCTL99)	byte	read	read/write	read/write	on page 1590
0x128	SENT_0 Control Register (ME_PCTL104)	byte	read	read/write	read/write	on page 1590
0x130	ADCSAR_b Control Register (ME_PCTL112)	byte	read	read/write	read/write	on page 1590
0x13B	ADCSAR_4 Control Register (ME_PCTL123)	byte	read	read/write	read/write	on page 1590
0x13F	ADCSAR_0 Control Register (ME_PCTL127)	byte	read	read/write	read/write	on page 1590
0x140	GTMINT Control Register (ME_PCTL128)	byte	read	read/write	read/write	on page 1590

Table 937. MC_ME register description(Continued)

Address	Name	Size	Access			Location
			User	Supervisor	Test	
0x1C0	Core Status Register (ME_CS)	word	read	read	read	*on page 1591*
0x1C4	core_0 Control Register (ME_CCTL0)	half word	read	read/write	read/write	*on page 1592*
0x1E0	core_0 Address Register (ME_CADDR0)	word	read	read/write	read/write	*on page 1593*

Note: Any access to unused registers as well as write accesses to read-only registers:

- *does not change register content*
- *causes a transfer error*

56.3.2 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.

56.3.2.1 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	1	0	0	S_PDO	0	0	S_MVR	0	0	S_FLA	
W																
Reset	0	0	0	0	1	1	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	0	S_PLL0	S_XOSC	S_IRC	S_SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 973. Global Status Register (ME_GS)

This register contains global mode status.

Table 938. Global Status Register (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 ***RUN*** 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
5–7 Reserved	
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 slew rate control mechanism of the output pads is disable to reduce power consumption, but the state of the output remains functional.
9–10 Reserved	
11 S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
12–13 Reserved	
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
16–24 Reserved	
25 S_PLL0	primary PLL status 0 primary PLL is not stable 1 primary PLL is providing a stable clock

Table 938. Global Status Register (ME_GS) field descriptions(Continued)

Field	Description
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_IRC	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 PLL0 PHI 0011 PLL0 PHI1 0100 – 1110 reserved 1111 system clock is disabled

56.3.2.2 Mode Control Register (ME_MCTL)

Address 0x004 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	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 974. 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 939. Mode Control Register (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)
4–15 Reserved	
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.** KEY:0101101011110000 (0x5AF0) INVERTED KEY:1010010100001111 (0xA50F)

56.3.2.3 Mode Enable Register (ME_ME)

Address 0x008 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	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 975. Mode Enable Register (ME_ME)

This register allows a way to 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 940. Mode Enable Register (ME_ME) field descriptions

Field	Description
0–15 Reserved	
16 RESET_DEST	‘destructive’ *RESET* mode enable 1 ‘destructive’ ***RESET*** mode is enabled
17–20 Reserved	
21 STOP0	*STOP0* mode enable 0 ***STOP0*** mode is disabled 1 ***STOP0*** mode is enabled
22 Reserved	
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

56.3.2.4 Interrupt Status Register (ME_IS)

Address 0x00C 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	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 976. Interrupt Status Register (ME_IS)

This register provides the current interrupt status.

Table 941. Interrupt Status Register (ME_IS) field descriptions

Field	Description
0–25 Reserved	
26 I_ICONF_CC	**Invalid mode configuration interrupt (core configuration)** — This bit is set if a write access to one of the **ME_CCTL*n*** registers is attempted while a mode transition is in progress. 0 No write to an **ME_CCTL*n*** register was attempted during an ongoing mode transition 1 A write to an **ME_CCTL*n*** 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 941. Interrupt Status Register (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

56.3.2.5 Interrupt Mask Register (ME_IM)

Address 0x010 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	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 977. Interrupt Mask Register (ME_IM)

This register controls whether an event generates an interrupt or not.

Table 942. Interrupt Mask Register (ME_IM) field descriptions

Field	Description
0–25 Reserved	
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

Table 942. Interrupt Mask Register (ME_IM) field descriptions(Continued)

Field	Description
29 M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
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

56.3.2.6 Invalid Mode Transition Status Register (ME_IMTS)

Address 0x014 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	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 978. Invalid Mode Transition Status Register (ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Table 943. Invalid Mode Transition Status Register (ME_IMTS) field descriptions

Field	Description
0–25 Reserved	
26 S_MRIG	Mode Request Ignored status This bit is set whenever a new mode is requested while a transition to the SAFE mode is in progress, a transition to STOP0 mode is requested while a wakeup or interrupt request is active, or a transition to HALT0 mode is requested while an interrupt request is active. It is cleared by writing a ‘1’ to this bit. 0 Mode transition requested is not ignored. 1 Mode transition requested is ignored.
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’). Refer to *Section 56.4.5: 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

Table 943. Invalid Mode Transition Status Register (ME_IMTS) field descriptions(Continued)

Field	Description
28 S_MRI	**Mode Request Illegal status** — This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a '1' to this bit. 0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode
29 S_DMA	**Disabled Mode Access status** — This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode
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

56.3.2.7 Debug Mode Transition Status Register (ME_DMTS)

Address 0x018 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	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	IRC_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 979. 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 944. Debug Mode Transition Status Register (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
4–7 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
9–10 Reserved	
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 944. Debug Mode Transition Status Register (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
16 Reserved	
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 IRC_SC	**IRC 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 IRC.) 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
22 Reserved	
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 944. Debug Mode Transition Status Register (ME_DMTS) field descriptions(Continued)

Field	Description
25 CDP_PRPH_192_223	**Clock Disable Process Pending status for Peripherals 192…2232** — This bit is set when any peripheral appearing in ME_PS6 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**[2] — This bit is set when any peripheral appearing in **ME_PS5** 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**[2] — 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**[2] — 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**[2] — 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
30 CDP_PRPH_32_63	**Clock Disable Process Pending status for Peripherals 32…63**[2] — 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**[2] — This bit is set when any peripheral appearing in **ME_PS0** has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

1. Peripheral n corresponds to the **ME_PCTL*n*** register. Refer to *Table 937* for the **ME_PCTL*n*** locations actually occupied, which in turn indicates which peripherals are reported in the **ME_DMTS** register.

56.3.2.8 *RESET* Mode Configuration Register (ME_RESET_MC)

Address 0x020 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	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	0	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 980. RESET Mode Configuration Register (ME_RESET_MC)

This register configures system behavior during *RESET* mode. Refer to *Table 945* for details.

Note: The following configuration values are set according to the chip configuration: XOSCON

56.3.2.9 *TEST* Mode Configuration Register (ME_TEST_MC)

Address 0x024 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	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	0	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 981. TEST Mode Configuration Register (ME_TEST_MC)

This register configures system behavior during *TEST* mode. Refer to *Table 945* for details.

Note: Byte write accesses are not allowed to this register.

56.3.2.10 *SAFE* Mode Configuration Register (ME_SAFE_MC)

Address 0x028 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	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	0	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 982. SAFE Mode Configuration Register (ME_SAFE_MC)

This register configures system behavior during ***SAFE*** mode. Refer to *Table 945* for details.

Note: Byte write accesses are not allowed to this register.

56.3.2.11 *DRUN* Mode Configuration Register (ME_DRUN_MC)

Address 0x02C 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	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	0	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 983. DRUN Mode Configuration Register (ME_DRUN_MC)

This register configures system behavior during ***DRUN*** mode. Refer to *Table 945* for details.

Note: Byte write accesses are not allowed to this register.

Note: The XOSCON configuration value is set according to the chip configuration:

56.3.2.12 *RUN0…3* Mode Configuration Registers (ME_RUN0…3_MC)

Address 0x030 - 0x03C 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	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	0	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 984. RUN0...3 Mode Configuration Registers (ME_RUN0...3_MC)

This register configures system behavior during ***RUN0…3*** modes. Refer to *Table 945* for details.

Note: Byte write accesses are not allowed to this register.

56.3.2.13 *HALT0* Mode Configuration Register (ME_HALT0_MC)

Address 0x040 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	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	0	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 985. HALT0 Mode Configuration Register (ME_HALT0_MC)

This register configures system behavior during ***HALT0*** mode. Refer to *Table 945* for details.

Note: Byte write accesses are not allowed to this register.

56.3.2.14 *STOP0* Mode Configuration Register (ME_STOP0_MC)

Address 0x048 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	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	0	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 986. STOP0 Mode Configuration Register (ME_STOP0_MC)

This register configures system behavior during *STOP0* mode. Refer to *Table 945* for details.

Note: Byte write accesses are not allowed to this register.

Table 945. Mode Configuration Registers (ME_<*mode*>_MC) field descriptions

Field	Description
0 Reserved	
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.
4–7 Reserved	
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.
9–10 Reserved	
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
12–13 Reserved	

Table 945. Mode Configuration Registers (ME_<*mode*>_MC) field descriptions(Continued)

Field	Description
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
16–24 Reserved	
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
27 IRCON	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 PLL0 PHI 0011 PLL0 PHI1 0100 – 1110 reserved 1111 system clock is disabled in ***TEST*** mode, reserved in all other modes

56.3.2.15 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	S_PIT_RTI_1	S_PIT_RTI_0	S_DECIFILTER	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	S_SIPI_0	0	S_LFAST_0	0	0	0	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 987. Peripheral Status Register 0 (ME_PS0)

This register provides the status of the peripherals. Refer to *Table 946* for details.

56.3.2.16 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	S_ADCSD_0	S_ADCSD_2	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	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 988. Peripheral Status Register 1 (ME_PS1)

This register provides the status of the peripherals. Refer to *Table 946* for details.

56.3.2.17 Peripheral Status Register 2 (ME_PS2)

Address 0x068 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	S_LINFlexD_0	S_LINFlexD_1	0	0	0	0	0	S_LINFlexD_14	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	S_CAN_RAM_CTRL	S_CCCU	0	0	S_MCAN_1	S_MCAN_2	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 989. Peripheral Status Register 2 (ME_PS2)

This register provides the status of the peripherals. Refer to *Table 946* for details.

56.3.2.18 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	0	0	0	S_ADCSAR_4	0	0	0	0	0	0	0	0	0	0	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	S_SENT_0	0	0	0	0	S_DSPI_0	0	S_DSPI_4	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 990. Peripheral Status Register 3 (ME_PS3)

This register provides the status of the peripherals. Refer to *Table 946* for details.

56.3.2.19 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	0	0	0	S_GTMINT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 991. Peripheral Status Register 4 (ME_PS4)

This register provides the status of the peripherals. Refer to *Table 946* for details.

Table 946. Peripheral Status Registers (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

56.3.2.20 Run Peripheral Configuration Registers (ME_RUN_PC0…7)

Address 0x080 - 0x09C 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	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 992. Run Peripheral Configuration Registers (ME_RUN_PC0…7)

These registers configure eight different types of peripheral behavior during run modes.

Table 947. Run Peripheral Configuration Registers (ME_RUN_PC0…7) field descriptions

Field	Description
0–23 Reserved	
24 RUN3	Peripheral control during *RUN3* 0 Peripheral is frozen with clock gated 1 Peripheral is active
25 RUN2	Peripheral control during *RUN2* 0 Peripheral is frozen with clock gated 1 Peripheral is active
26 RUN1	Peripheral control during *RUN1* 0 Peripheral is frozen with clock gated 1 Peripheral is active
27 RUN0	Peripheral control during *RUN0* 0 Peripheral is frozen with clock gated 1 Peripheral is active
28 DRUN	Peripheral control during *DRUN* 0 Peripheral is frozen with clock gated 1 Peripheral is active
29 SAFE	Peripheral control during *SAFE* 0 Peripheral is frozen with clock gated 1 Peripheral is active
30 TEST	Peripheral control during *TEST* 0 Peripheral is frozen with clock gated 1 Peripheral is active
31 RESET	Peripheral control during *RESET* 0 Peripheral is frozen with clock gated 1 Peripheral is active

56.3.2.21 Low-Power Peripheral Configuration Registers (ME_LP_PC0…7)

Address 0x0A0 - 0x0BC 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	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 993. Low-Power Peripheral Configuration Registers (ME_LP_PC0…7)

These registers configure eight different types of peripheral behavior during non-run modes.

Table 948. Low-Power Peripheral Configuration Registers (ME_LP_PC0…7) field descriptions

Field	Description
0–20 Reserved	
21 STOP0	Peripheral control during *STOP0* 0 Peripheral is frozen with clock gated 1 Peripheral is active
22 Reserved	
23 HALT0	Peripheral control during *HALT0* 0 Peripheral is frozen with clock gated 1 Peripheral is active
24–31 Reserved	

56.3.2.22 Peripheral Control Registers (ME_PCTL*n*)

Address 0x0C0 - 0x1BF Access: User read, 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 994. Peripheral Control Registers (ME_PCTLn)

These registers select the configurations during run and non-run modes for each peripheral. Refer to *Table 937* for information on which **ME_PCTLn** locations are actually occupied. The unoccupied locations contain a read-only byte value of 0x00.

Table 949. Peripheral Control Registers (ME_PCTLn) field descriptions

Field	Description
0 Reserved	
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 949. Peripheral Control Registers (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

56.3.2.23 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 995. Core Status Register (ME_CS)

This register provides the status of each core.

Table 950. Core Status Register (ME_CS) field descriptions

Field	Description
0–30 Reserved	
31 S_CORE0	**Core status 0** — This bits specifies the current status of core_0 which is controlled by the **ME_CCTL0** register. 0 core_0 is disabled 1 core_0 is running

56.3.2.24 Core Control Register (ME_CCTL0)

Address 0x1C4 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	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 996. Core Control Register (ME_CCTL0)

This register controls whether 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 (i.e., while the **S_MTRANS** bit of the **ME_GS** 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.

Table 951. Core Control Register (ME_CCTL0) field descriptions

Field	Description
0–3 Reserved	
5 STOP0	**Core control during *STOP0*** — core_0 is always disabled during ***STOP0***.
6 Reserved	
7 HALT0	**Core control during *HALT0*** — core_0 is always disabled during ***HALT0***.
8 RUN3	**Core control during *RUN3*** 0 core_0 is disabled with clock gated 1 core_0 is running
9 RUN2	**Core control during *RUN2*** 0 core_0 is disabled with clock gated 1 core_0 is running

Table 951. Core Control Register (ME_CCTL0) field descriptions(Continued)

Field	Description
10 RUN1	**Core control during *RUN1*** 0 core_0 is disabled with clock gated 1 core_0 is running
11 RUN0	**Core control during *RUN0*** 0 core_0 is frozen with clock gated 1 core_0 is running
12 DRUN	**Core control during *DRUN*** 0 core_0 is frozen with clock gated 1 core_0 is running
13 SAFE	**Core control during *SAFE*** 0 core_0 is frozen with clock gated 1 core_0 is running
14 TEST	**Core control during *TEST*** 0 core_0 is frozen with clock gated 1core_0 is running
15 RESET	**Core control during *RESET*** — core_0 is always disabled during ***RESET***.

56.3.2.25 Core Control Registers (ME_CADDR0)

Address 0x1E0 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 W	ADDR[31:16]															
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[15:2]														0	RMC
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0

Figure 997. Core Control Registers (ME_CADDR0)

This register gives the boot address for core_0 and a bit for controlling whether core_0 is to be reset on the next mode change that has 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 (i.e., while the **S_MTRANS** bit of the **ME_GS** 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 **ADDR** field of the **ME_CADDR0** is determined by the chip configuration and boot mode.*

Table 952. Core Control Registers (ME_CADDR0) field descriptions

Field	Description
0–29 ADDR	**Core Address** — This field is used by core_0 as the boot address (32-bit word aligned) when core_0 next exits reset.
30 Reserved	
31 RMC	**Reset on Mode Change** — core_0 will be reset on the next mode change that has core_0 configured to be running in the target mode. 0 core_0 will not be reset on the next mode change 1 core_0 will be reset on the next mode change that has core_0 configured to be running in the target mode

56.4 Functional Description

56.4.1 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 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

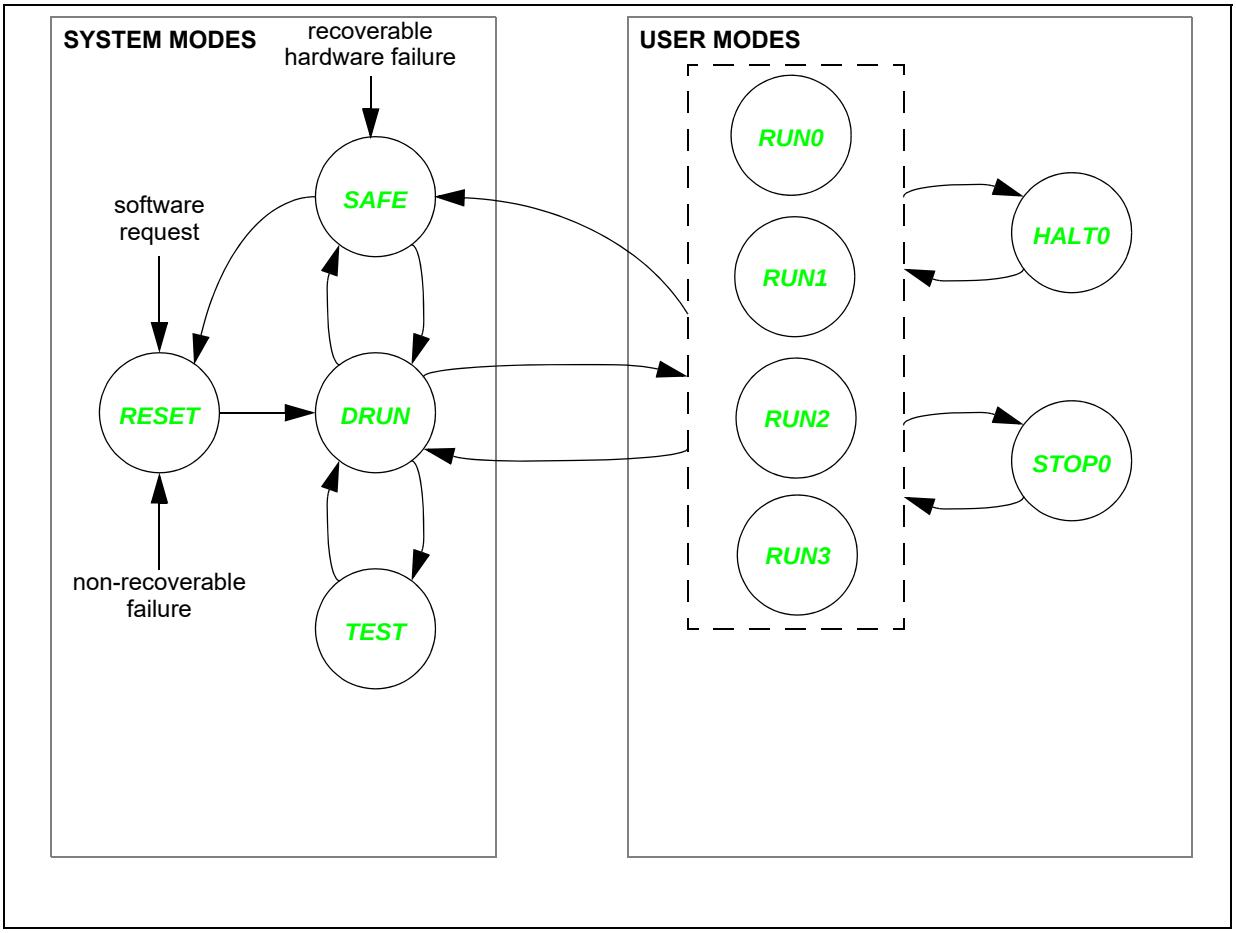
- the first time with the value of the key (0x5AF0) into the **KEY** bit field and the required target mode into the **TARGET_MODE** bit field,
- and the second time with the inverted value of the key (0xA50F) into the **KEY** bit field and the required target mode into the **TARGET_MODE** bit field.

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, refer to *Section 56.4.5, "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*** → ***RUN0...3***, ***DRUN*** → ***DRUN***, ***SAFE*** → ***SAFE***, and ***TEST*** → ***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 the **S_MTRANS** bit in the **ME_GS** register after requesting a transition to **HALT0** or **STOP0** modes.*

Figure 998. MC_ME Mode Diagram



56.4.2 Modes Details

56.4.2.1 *RESET* Mode

The chip enters this mode on the following events:

- from ***SAFE***, ***DRUN***, ***RUN0...3***, or ***TEST*** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register 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 pre-defined configuration, and the 16 MHz internal RC oscillator is selected as the system clock.

56.4.2.2 *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 **TARGET_MODE** bit field of the **ME_MCTL** register 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.

56.4.2.3 *SAFE* Mode

The chip enters this mode on the following events:

- from any mode except ***RESET*** when the **TARGET_MODE** bit field of the **ME_MCTL** register 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 **PDO** bit of the **ME_SAFE_MC** register should be set. The input levels remain unchanged.

56.4.2.4 *TEST* Mode

The chip enters this mode on the following event:

- from the ***DRUN*** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “0001”

As soon as the above event 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 intended to be 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.

56.4.2.5 *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 **TARGET_MODE** bit field of the **ME_MCTL** register 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.

56.4.2.6 *HALT0* Mode

The chip enters this mode from one of the ***RUN0…3*** modes when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “1000”.

As soon as the above event 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 the additional core clocks frozen
- only a few peripherals running

It is intended to be used by software to wait until it is required to do something and then to react quickly (i.e., within a few system clock cycles of an interrupt event).

*Note: It is good practice for software to ensure that the **S_MTRANS** bit in the **ME_GS** register has been cleared on **HALT0** mode exit to ensure that the previous **RUN0...3** mode configuration has been fully restored before executing critical code.*

56.4.2.7 *STOP0* Mode

The chip enters this mode from one of the ***RUN0...3*** modes when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “1010”.

As soon as the above event 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
- the additional core clocks frozen
- almost all peripherals stopped

It is intended to be used by software to wait until it is required to do something with no need to react quickly (e.g., allow for system clock source to be re-started).

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 the **S_MTRANS** bit in the **ME_GS** register has been cleared on **STOP0** mode exit to ensure that the previous **RUN0...3** mode configuration has been fully restored before executing critical code.*

56.4.3 Mode Transition Process

The process of mode transition follows the below mentioned steps in a pre-defined 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.

56.4.3.1 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 pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the **TARGET_MODE** bit field is not updated. An optional interrupt can be generated for invalid mode requests. Refer to *Section 56.4.5: 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 **TARGET_MODE** bit field of the **ME_MCTL** register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit **S_MTRANS** of the **ME_GS** register.

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.

56.4.3.2 Target Mode Configuration Loading

On completion of the *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 . A '√' indicates that a given resource is configurable for a given mode.

Table 953. MC_ME Resource Control Overview

Resource	Mode						
	RESET	TEST	SAFE	DRUN	RUN0…3	HALT0	STOP0
IRC	on	√ on	on	on	on	on	on
XOSC	off	√ off	off	√ off	√ off	√ off	√ off
PLL0	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

56.4.3.3 Peripheral Clocks Disable

On completion of the *Target Mode Request* step and the system clock frequency ramp-down of the *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_PCTL*n***

Note: The MC_ME automatically requests peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. However, it is good practice for software to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a ***SAFE*** mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The ***SAFE*** mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Refer to *Section 56.4.6, "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.

56.4.3.4 Processor Low-Power Mode Entry

If, on completion of the *Peripheral Clocks Disable* step, the mode transition is to the ***HALT0***,***STOP0*** mode, the MC_ME requests each processor to enter its stopped state. Each processor acknowledges its stop state request bit after completing all outstanding bus transactions.

If, on completion of the *Peripheral Clocks Disable* step, the mode transition is from a non-low-power mode to another non-low-power mode, the MC_ME requests each processor which is configured in the **ME_CCTL*n*** registers to be disabled to enter its stopped state. The processor acknowledges its stop state request bit after completing all outstanding bus transactions.

56.4.3.5 Processor Clocks Disable

If, on completion of the *Processor Low-Power Mode Entry* step, and all processors which are configured in the **ME_CCTL*n*** registers 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 processors 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.

Warning: Clocks to the whole chip including the processor and system memory can be disabled in TEST mode.

56.4.3.6 System Clock Disable

If, on completion of the *Processor Clocks Disable* step, all processor clocks have been disabled, the system clock used for non-processor system functions and memory is disabled.

56.4.3.7 Clock Sources Switch-On

On completion of the *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 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.

56.4.3.8 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 **S_FLA** bit field of the **ME_GS** register 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.

56.4.3.9 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 slew rate control mechanism is switched on

56.4.3.10 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.

56.4.3.11 System and Processor Clocks Enable

On completion of the *Flash Module Switch-On* step, the MC_ME enables the non-processor system clock and the clocks of all processors which are configured in the **ME_CCTL*n*** registers to be disabled in the current mode and to be running in the target mode.

The MC_ME initiates the resetting of all processors which are configured in the **ME_CCTL*n*** registers to be running in the target mode and which have their **RMC** bits in the **ME_CADDR*n*** registers set to '1'. The core_rst_b outputs remain asserted until the end of the mode transition as indicated by the **S_MTRANS** bit of the **MW_GS** register 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 *System Clock Switching* process is completed.

56.4.3.12 Processor Low-Power Mode Exit

On completion of the *System and Processor Clocks Enable* step, the MC_ME requests all processors which are configured in the **ME_CCTL*n*** registers to be disabled in the current mode and to be running in the target mode.

56.4.3.13 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 **S_IRC** bit of the **ME_GS** register is set by hardware (i.e., the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the external crystal oscillator takes effect only after the **S_XOSC** bit of the **ME_GS** register is set by hardware (i.e., the external crystal oscillator has stabilized).
- The target clock configuration for the PLL0 PHI takes effect only after the **S_PLL0** bit of the **ME_GS** register is set by hardware (i.e., the primary PLL has stabilized).
- To disable the clock, the **SYSCLK** bit field should be programmed with "1111". This is only possible in ***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 **S_SYSCLK** bit field of the **ME_GS** register, 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 *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 *Peripheral Clocks Disable* process has completed in order not to change the system clock frequency before peripherals close their internal activities the *Peripheral Clocks Enable* process has completed.

An overview of system clock source selection possibilities for each mode is shown in *Table 954*. A '√' indicates that a given clock source is selectable for a given mode.

Table 954. MC_ME System Clock Selection Overview

System clock source	Mode						
	RESET	TEST	SAFE	DRUN	RUN0…3	HALT0	STOP0
16 MHz internal RC oscillator	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)
external crystal oscillator		√		√	√	√	√
PLL0 PHI		√		√	√	√	
system clock is disabled		√(1)					

1. disabling the system clock during ***TEST*** mode will require a reset in order to exit ***TEST*** mode

56.4.3.14 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 *Peripheral Clocks Disable* process has completed.

56.4.3.15 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 switched off at this step:

- the 16 MHz internal RC oscillator
- the external crystal oscillator
- the primary PLL

For the clock sources related to the system clock, this step is executed only after the *System Clock Switching* process has completed.

56.4.3.16 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 **S_FLA** bit field of the **ME_GS** register. This step is executed only when the *Processor Clocks Disable* and *System Clock Disable* processes have completed.

56.4.3.17 Current Mode Update

The current mode status bit field **S_CURRENT_MODE** of the **ME_GS** register is updated with the target mode bit field **TARGET_MODE** of the **ME_MCTL** register when :

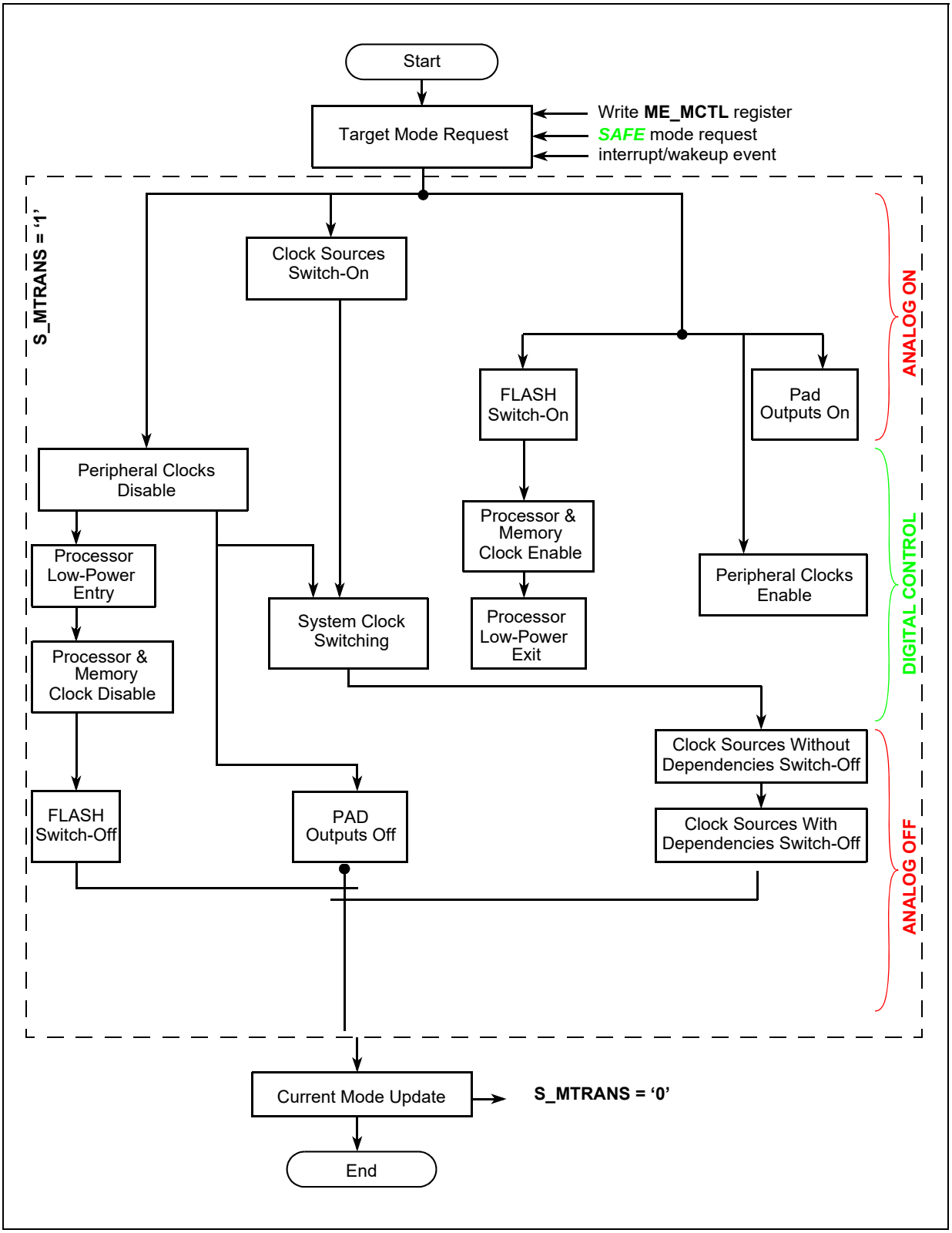
- all the updated status bits in the **ME_GS** register match the configuration specified in the **ME_<target mode>_MC** register
- power sequences are done
- clock disable/enable process is finished
- processor low-power mode (stop) entry and exit processes are finished

*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 **S_MTRANS** bit of the **ME_GS** register. 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 it is expected, the **ME_DMTS** register can indicate which process is still in progress.

Figure 999. MC_ME Transition Diagram



56.4.4 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, IRC must be on.
- If the external crystal oscillator clock is selected as the system clock, XOSC must be on.
- If the PLL0 PHI clock is selected as the system clock, PLL0 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.

56.4.5 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.

56.4.5.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the **ME_<mode>_MC** registers violating the protection rules mentioned in the *Section 56.4.4: Protection of Mode Configuration Registers*, the interrupt pending bit **I_ICONF** of the **ME_IS** register is set and an interrupt request is generated if the mask bit **M_ICONF** of the **ME_IM** register 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 **I_ICONF_CU** of the **ME_IS** register is set and an interrupt request is generated if the mask bit **M_ICONF_CU** of the **ME_IM** register is ‘1’.

If an attempt to write to one of the **ME_CCTLn** or **ME_CADDRn** 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 **S_MTRANS** bit in the **ME_GS** register deasserting, the interrupt pending bit **I_ICONF_CC** of the **ME_IS** register is set and an interrupt request is generated if the mask bit **M_ICONF_CC** of the **ME_IM** register is ‘1’.

56.4.5.2 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 **S_SEA** bit of the **ME_IMTS** register is set.
- If the **TARGET_MODE** bit field of the **ME_MCTL** register is written with a value different from the specified mode values (i.e., a non-existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the **S_NMA** bit of the **ME_IMTS** register 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 **S_DMA** bit of the **ME_IMTS** register 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 **S_MRI** of the **ME_IMTS** register 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 **S_MTRANS** bit of the **ME_GS** register is '1'), the mode transition illegal status bit **S_MTI** of the **ME_IMTS** register 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 (e.g., ***RUN0*** → ***RUN0***) are not considered as invalid even when the mode transition process is active (i.e., **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 (i.e., 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 **I_IMODE** of the **ME_IS** register is set, and an interrupt request is generated if the mask bit **M_IMODE** of the **ME_IM** register is '1'.

56.4.5.3 *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 **I_SAFE** of the **ME_IS** register is set, and an interrupt is generated if the mask bit **M_SAFE** of **ME_IM** register 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 (i.e., programming of **ME_MCTL** register).

56.4.5.4 Mode Transition Complete interrupt

Whenever the system fully completes a mode transition (i.e., the **S_MTRANS** bit of **ME_GS** register transits from '1' to '0'), the interrupt pending bit **I_MTC** of the **ME_IS** register is set, and an interrupt request is generated if the mask bit **M_MTC** of the **ME_IM** register 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.

56.4.6 Peripheral Clock Gating

During all chip modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers **ME_RUN_PC0…7** are chosen only during the software running modes ***DRUN***, ***TEST***, ***SAFE***, and ***RUN0…3***. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the **RUN_CFG** bit field of the **ME_PCTL*n*** registers.

The low-power peripheral configuration registers **ME_LP_PC0…7** are chosen only during the low-power modes ***HALT0*** and ***STOP0***. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the **LP_CFG** bit field of the **ME_PCTL*n*** registers.

Any modifications to the **ME_RUN_PC0…7**, **ME_LP_PC0…7**, and **ME_PCTL*n*** registers do not affect the clock gating behavior until a new mode transition request is generated.

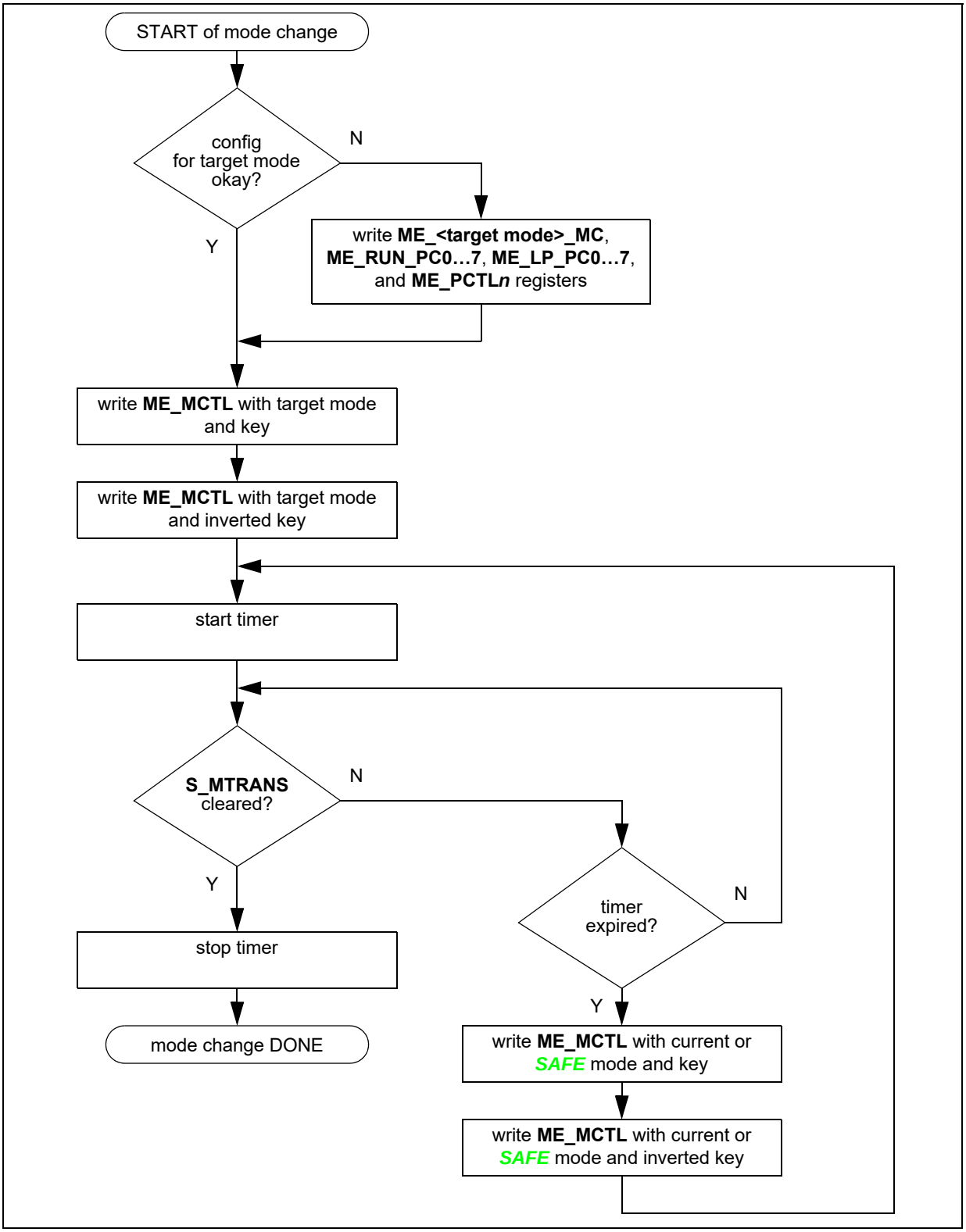
Whenever the chip enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the **DBG_F** bit of the associated **ME_PCTL*n*** register is set. Otherwise, the peripheral clock gating status depends on the **RUN_CFG** and **LP_CFG** bits.

56.4.7 Application example

Figure 1000 shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

Figure 1000. MC_ME application example flow diagram



57 e200z215An3 Core Debug Support

This chapter describes the debug features of the e200z215An3 core.

57.1 Overview

Internal debug support in the e200z215An3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware-based debugger, which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of PowerISA 2.06. e200z215An3 supports a subset of these defined facilities. In addition to the facilities defined in PowerISA 2.06, e200z215An3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

57.1.1 Software debug facilities

e200z215An3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is “owned” by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

57.1.1.1 PowerISA 2.06 compatibility

The e200z215An3 core implements a subset of the PowerISA 2.06 internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

57.1.2 Additional debug facilities

In addition to the debug functionality defined in PowerISA 2.06, e200z215An3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See *Section 57.1.4, Sharing debug resources by software/hardware*.) A data trace port is also provided.

57.1.2.1 Data trace port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z215An3 core. This port provides information corresponding to each read or write access completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (i.e. the normal setting. See the “Data Trace Control Register (DTC)” section in the Core (e200z215An3) Nexus 3 Module chapter. The data trace port will report aligned accesses and misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0.

57.1.3 Hardware debug facilities

The e200z215An3 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 e200z215An3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the “hardware-owned” debug facilities. In addition, since resources are “owned” by the hardware debugger, inconsistent values may be present if software attempts to read “hardware-owned” debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in *Section 57.3, Debug registers* are reserved for use by the external debugger. The same events described in *Section 57.2, Software debug events and exceptions* are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via

EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

57.1.4 Sharing debug resources by software/hardware

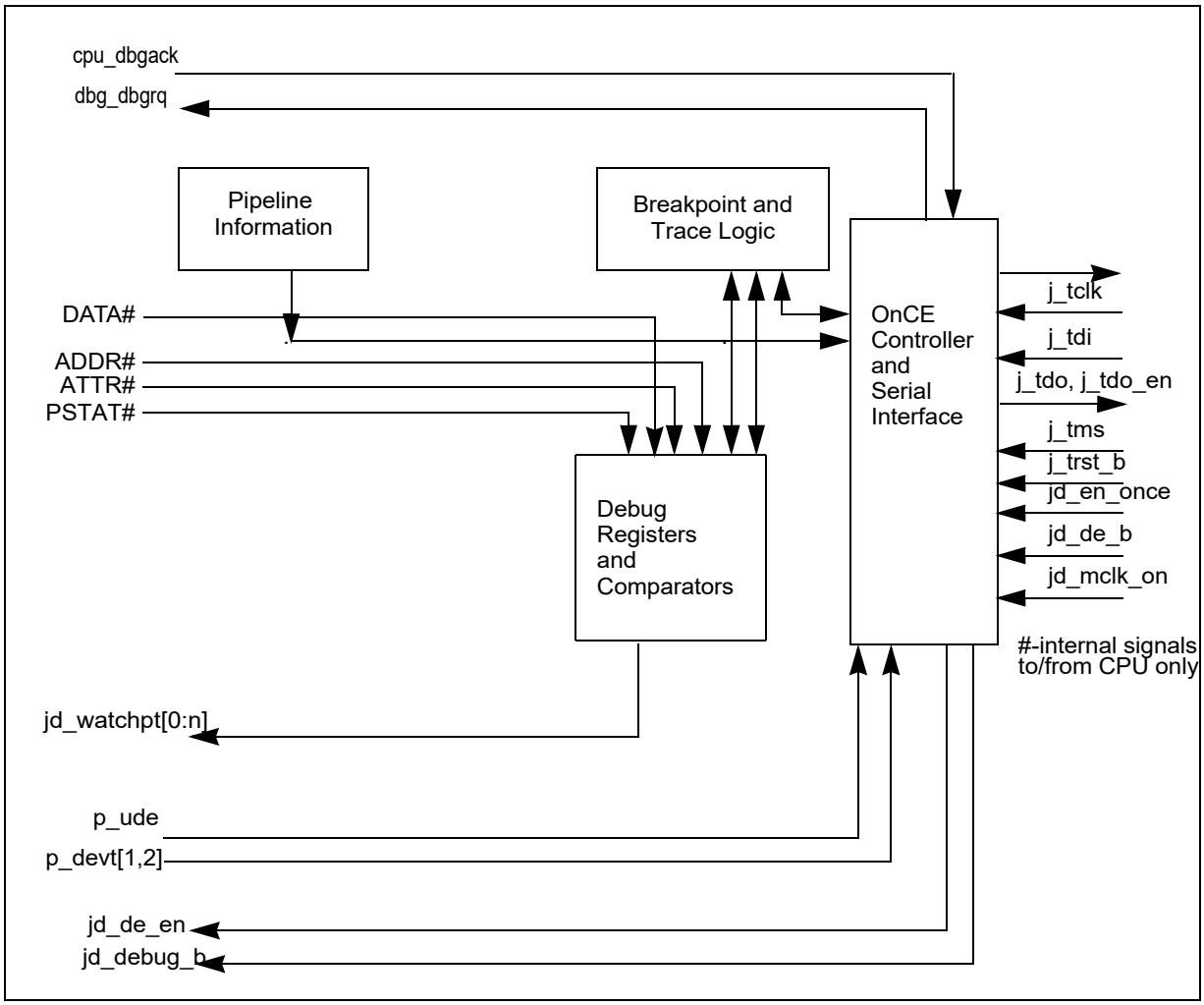
Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources that set DBSR bits when $DBCR0_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources that set EDBSR0 bits when $EDBCR0_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

57.1.4.1 Simultaneous hardware and software debug event handing

Since it is possible that a “hardware-owned” resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented that guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

Figure 1001 shows the e200z215An3 debug resources.

Figure 1001. e200z215An3 debug resources



57.2 Software debug events and exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$=1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource that is “owned” by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may

cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction that enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** that 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 PowerISA 2.06:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z215An3 defines additional debug events:

- The External debug events DEVT1 and DEVT2, which are described in *Section 57.2.11, External debug event*.
- The Performance Monitor Interrupt event PMI, which is described in *Section 57.2.13, Performance Monitor Interrupt debug event*.

The e200z215An3 debug configuration supports most of these event types. Unsupported PowerISA 2.06 defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

57.2.1 Instruction Address Compare event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

57.2.2 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, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load or store operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note: In contrast to the PowerISA 2.06 definition, Data Address Compare events on e200z215An3 do not prevent the load or store class instruction from completing. If a load or store class instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note: DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

Note: DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

*Note: DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.*

*Note: Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.*

57.2.2.1 Data Address Compare event status updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare

following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC_OFST}$ field when the DVC DAC status is recorded.

Table 955 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0.

Table 955. DAC events and resultant updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

Table 955. DAC events and resultant updates(Continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	Ldst instruction, no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DVC DACy, Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	Any instruction (no exception)	DSI, with or without DAC, Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction (no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

57.2.3 Linked Instruction Address Compare and Data Address Compare events

Data Address Compare 1, 2, 3, and 4 debug events may be ‘linked’ with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1 may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC ‘hit’ also generates a corresponding linked IAC ‘hit’. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note: Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

57.2.4 Trap debug event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

57.2.5 Branch Taken debug event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ cannot be set by a Branch Taken debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

57.2.6 Instruction Complete debug event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the

instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction that generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note: Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}$=1, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while MSR_{DE}=1.

57.2.7 Interrupt Taken debug event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

57.2.8 Critical Interrupt Taken debug event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

57.2.9 Return debug event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (i.e. before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

57.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (i.e. before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

57.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to ‘1’ to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

57.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to ‘1’ to record the debug exception. This debug event is an asynchronous event.

57.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to ‘1’ to record the debug exception. This debug event is an asynchronous event.

57.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not “owned” by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not “owned” by software will not cause

modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not “owned” by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

57.3.1 Debug Address and Value registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in *Figure 1002*) corresponds to the byte data value transferred to/from memory byte offset 0, 8, ..., and the least significant byte of the register (labeled B7 in *Figure 1002*) corresponds to byte offset 7, F, ... When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison; they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z215An3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

B0								B1								B2								B3							
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
B4								B5								B6								B7							
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

SPR 601 (DVC1U), 318 (DVC1), 602 (DVC2U) 319 (DVC2); Read/Write; Reset: Unaffected

Figure 1002. DVC1, DVC2 registers

57.3.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z215An3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

57.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z215An3 adds some implementation specific bits to this register, as seen in *Figure 1003*.

EDM	IDM	RST	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC2	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	0	CIRPT	CRET	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 308; Read/Write; Reset(1) - 0x0

Figure 1003. Debug Control Register 0 (DBCR0) register

1. $DBCR0_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if $DBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $DBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b,** and only software-owned resources indicated by EDBRAC0 and the $DBCR0_{RST}$ field will be reset by **p_reset_b**. The $DBCR0_{RST}$ field will always be reset by **p_reset_b** regardless of the value of $DBCR0_{EDM}$.

Table 956 provides bit definitions for Debug Control Register 0.

Table 956. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0.
		Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE}=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.
2:3	RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled 1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled

Table 956. DBCR0 field descriptions (Continued)

Bit	Name	Description
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	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1 IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled

Table 956. DBCR0 field descriptions (Continued)

Bit	Name	Description
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

57.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in *Figure 1004*.

IAC1US	IAC1ER	IAC2US	IAC2ER	IAC12M	0	IAC3US	IAC3ER	IAC4US	IAC4ER	IAC34M	0
0 1	2 3	4 5	6 7	8 9	10 11 12 13 14 15	16 17	18 19	20 21	22 23	24 25	26 27 28 29 30 31

SPR - 309; Read/Write; Reset(1) - 0x0

Figure 1004. DBCR1 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b.**

Table 957 provides bit definitions for Debug Control Register 1.

Table 957. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode 00 IAC1 debug events not affected by MSR_{PR} 01 Reserved 10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode 00 IAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode 00 IAC2 debug events not affected by MSR_{PR} 01 Reserved 10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode 00 IAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1

Table 957. DBCR1 field descriptions (Continued)

Bit	Name	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 (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 (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

57.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in *Figure 1005*.

DAC1US	DAC1ER	DAC2US	DAC2ER	DAC12M	DAC1LNK	DAC2LNK	DVC1M	DVC2M	DVC1BE	DVC2BE
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

SPR - 310; Read/Write; Reset(1) - 0x0

Figure 1005. DBCR2 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 958 provides bit definitions for Debug Control Register 2.

Table 958. DBCR2 field descriptions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1

Table 958. DBCR2 field descriptions (Continued)

Bit	Name	Description
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies *Exact Address Compare* since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR.

Table 958. DBCR2 field descriptions (Continued)

Bit	Name	Description
12:13	DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **Note:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **Note:** Inactive byte lanes of the memory access are automatically masked.
14:15	DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **Note:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **Note:** Inactive byte lanes of the memory access are automatically masked.

Table 958. DBCR2 field descriptions (Continued)

Bit	Name	Description
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1*xxxxxxx* Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. *x*1*xxxxxx* Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. *xx*1*xxxxx* Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. *xxx*1*xxxx* Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. *xxxx*1*xxx* Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. *xxxxx*1*xx* Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. *xxxxxx*1*x* Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. *xxxxxxx*1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1*xxxxxxx* Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. *x*1*xxxxxx* Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. *xx*1*xxxxx* Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. *xxx*1*xxxx* Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. *xxxx*1*xxx* Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. *xxxxx*1*xx* Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2. *xxxxxx*1*x* Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. *xxxxxxx*1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

57.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in *Figure 1006*.

0	DVC1C	0	DVC2C	0	0	DAC1XMH	0	DAC2XMH	DAC1XM	DAC2XM	DAC1CFG	DAC2CFG
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

SPR - 563; Read/Write; Reset(1) - 0x0

Figure 1006. DBCR4 register

1. DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 959 provides bit definitions for Debug Control Register 4.

Table 959. DBCR4 field descriptions

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. 1 Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition 0 Normal DVC2 operation. 1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field 0 DAC1XM masks 0–15 low-order address bits 1 DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.

Table 959. DBCR4 field descriptions (Continued)

Bit	Name	Description
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$=00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$=00 0011 Reserved 01*xx* Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). **Note:** Unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01*xx* Reserved 1*xxx* Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **Note:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

57.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in *Figure 1007*

IAC5US	IAC5ER	IAC6US	IAC6ER	IAC56M	0	IAC7US	IAC7ER	IAC8US	IAC8ER	IAC78M	0
0 1	2 3	4 5	6 7	8 9	10 11 12 13 14 15	16 17	18 19	20 21	22 23	24 25	26 27 28 29 30 31

SPR - 564; Read/Write; Reset(1) - 0x0

Figure 1007. DBCR5 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 960 provides bit definitions for Debug Control Register 5.

Table 960. DBCR5 field descriptions

Bit	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved

Table 960. DBCR5 field descriptions (Continued)

Bit	Name	Description
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved 10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

57.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in *Figure 1008*.

IAC1XM	IAC2XM	IAC3XM	IAC4XM	IAC5XM	IAC6XM	IAC7XM	IAC8XM

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

SPR - 603; Read/Write; Reset(1) - 0x0

Figure 1008. DBCR6 register

1. DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 961 provides bit definitions for Debug Control Register 6.

Table 961. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 = Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Control. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

Table 961. DBCR6 field descriptions (Continued)

Bit	Name	Description
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101 – 1111 = Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

57.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in *Figure 1009*.

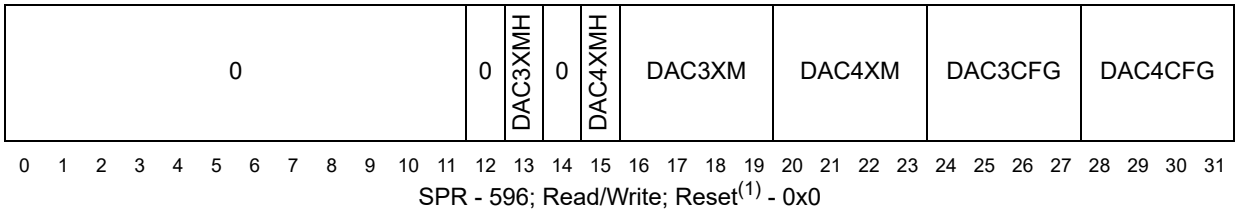


Figure 1009. DBCR7 register

1. DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 962 provides bit definitions for Debug Control Register 7.

Table 962. DBCR7 field descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High. DAC3XMH extends the range of the DAC3XM field. 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits

Table 962. DBCR7 field descriptions (Continued)

Bit	Name	Description
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001–1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

57.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in *Figure 1010*.

SPR - 597; Read/Write; Reset(1) - 0x0

Figure 1010. DBCR8 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 963 provides bit definitions for Debug Control Register 8.

Table 963. DBCR8 field descriptions

Bit	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 DAC3 debug events not affected by MSR_{PR} 01 Reserved 10 DAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC3 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 DAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC3 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC3 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 DAC4 debug events not affected by MSR_{PR} 01 Reserved 10 DAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC4 debug events can only occur if MSR_{PR}=1 (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 DAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC4 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1

Table 963. DBCR8 field descriptions (Continued)

Bit	Name	Description
8:9	DAC34M	Data Address Compare 3/4 Mode 00 Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 No effect 1 DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 No effect 1 DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

57.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr** *DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A ‘1’ causes the bit to be cleared, and a ‘0’ has no effect. Debug Status bits are set by Debug events only while Internal Debug mode is enabled ($DBCR0_{IDM}$=1). When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1, $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in *Figure 1011*.

Figure 1011. Debug Status Register (DBSR)

SPR - 304; Read/Clear; Reset - 0x1000_0000

Table 964 provides bit definitions for the Debug Status Register.

Table 964. DBSR field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR_{DE}=0 and $DBCR0_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store that is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 No reset occurred since these bits were last cleared by software 01 A hard reset occurred since these bits were last cleared by software 10 Reserved 11 Reserved
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	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a Return debug event occurred

Table 964. DBSR field descriptions (Continued)

Bit	Name	Description
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred.
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred.
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1.
24	—	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 PowerISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3‘b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3‘b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

57.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC). DDEAR is update by hardware with the effective address of the load or store operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in *Figure 1012*.

Data Effective Address

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

SPR - 600; Read/Write; Reset - unaffected

Figure 1012. Debug Data Effective Address Register (DDEAR)

57.3.3 External Debug Resource Allocation Control (EDBRAC0) register

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to ‘1’. EDBRAC0 provides a mechanism for the

hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$=1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$=0.

Hardware-owned resources that generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources that generate debug events if $DBCR0_{IDM}$=1 update DBSR, causing debug interrupts to occur if MSR_{DE}=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug mode is enabled. When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event that is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug mode is enabled ($EDBCR0_{EDM}$=1). When $EDBCR0_{EDM}$=1, a set bit in EDBSR0 by an event that is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in *Figure 1013*.

Figure 1013. External Debug Resource Allocation Control (EDBRAC0) register

0	IDM	RST	UDE	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC34	DAC2	0	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	PMI	0	CIRPT	CRET	DNI	DQM	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b**, reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Table 964 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 965. EDBRAC0 field descriptions

Bit	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control. When $EDBRAC0_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored. 0 Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable.
2	RST	Reset Field Control 0 $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 Event owned by hardware debug. 1 Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 Event owned by software debug. $DBCR0_{TRAP}$ is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 Event owned by software debug. IAC1 control fields are software readable/writable.
9	IAC2	Instruction Address Compare 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 Event owned by software debug. IAC4 control fields are software readable/writable.

Table 965. EDBRAC0 field descriptions (Continued)

Bit	Name	Description
12	DAC1	Data Address Compare 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields. 1 Event owned by software debug. DAC1 control fields are software readable/writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 Event owned by software debug. DAC2 control fields are software readable/writable.
15	—	Reserved
16	RET	Return Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field. 1 Event owned by software debug. $DBCR0_{RET}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 Event owned by software debug. IAC8 control are software readable/writable.
21	DEVT1	External Debug Event Input 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field. 1 Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field. 1 Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event 0 Event owned by hardware debug. No **mtpmr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 Event owned by software debug. PMRs are software readable/writable. **Note:** This bit is reset to ‘1’.
24	—	Reserved

Table 965. EDBRAC0 field descriptions (Continued)

Bit	Name	Description
25	CIRPT	Critical Interrupt Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field. 1 Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field. 1 Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$=1 and MSR_{DE}=1) or a nop ($DBCR0_{IDM}$=0 or MSR_{DE}=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENT_{DQTAG}$ field or DDAM register. Attempted access by software is ignored. 1 $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 966 shows which resources are controlled by EDBRAC0 settings.

Table 966. EDBRAC0 resource control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$		$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	**Software Accessible via mtspr, affected by p_reset_b**
0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	All debug registers
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{IDM}$
1	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{RST}$

Table 966. EDBRAC0 resource control(Continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IBM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$		$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DCM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{UDE}$
1	1	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{ICMP}$
1	1	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{BRT}$
1	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{IRPT}$
1	1	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{TRAP}$
1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US,IAC1ER}$, $DBCR6_{IAC1XM}$
1	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US,IAC2ER}$, $DBCR6_{IAC2XM}$
1	1	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR1_{IAC12M}$
1	1	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC3, $DBCR0_{IAC3}$, $DBCR1_{IAC3US,IAC3ER}$, $DBCR6_{IAC3XM}$

Table 966. EDBRAC0 resource control(Continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IBM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DCM}$	**Software Accessible via mtspr, affected by p_reset_b**
1	1	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC4, $DBCR0_{IAC4}$, $DBCR1_{IAC4US,IAC4ER}$, $DBCR6_{IAC4XM}$
1	1	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR1_{IAC34M}$
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC5, $DBCR0_{IAC5}$, $DBCR5_{IAC5US,IAC5ER}$, $DBCR6_{IAC5XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC6, $DBCR0_{IAC6}$, $DBCR5_{IAC6US,IAC6ER}$, $DBCR6_{IAC6XM}$
1	1	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR5_{IAC56M}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC7, $DBCR0_{IAC7}$, $DBCR5_{IAC7US,IAC7ER}$, $DBCR6_{IAC7XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC8, $DBCR0_{IAC8}$, $DBCR5_{IAC8US,IAC8ER}$, $DBCR6_{IAC8XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR5_{IAC78M}$

Table 966. EDBRAC0 resource control(Continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IBM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$		$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DCM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	DAC1, DVC1, DVC1U $DBCR0_{DAC1}$, $DBCR2_{DAC1US,DAC1ER}$, $DBCR2_{DVC1M,DVC1BE}$ $DBCR4_{DVC1C,DAC1XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	DAC3, DAC4 $DBCR7_{DAC\{3,4\}}$, $DAC\{3,4\}CFG$, $DAC\{3,4\}XM$, $DAC\{3,4\}XMH$ $DBCR8_{DAC\{3,4\}US}$, $DAC\{3,4\}ER$, $DAC\{3,4\}M$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	DAC2, DVC2, DVC2U $DBCR0_{DAC2}$, $DBCR2_{DAC2US,DAC2ER}$, $DBCR2_{DVC2M,DVC2BE}$ $DBCR4_{DVC2C,DAC2XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	1	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC12M}$
1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC1LNK}$
1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC2LNK}$

Table 966. EDBRAC0 resource control(Continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DDAM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	$DBCR8_{DAC3LNK}$
1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	1	—	—	—	—	—	—	—	—	—	—	$DBCR8_{DAC4LNK}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	$DBCR0_{RET}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	$DBCR0_{DEVT1}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	$DBCR0_{DEVT2}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	All PMRs
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	$DBCR0_{CIRPT}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	$DBCR0_{CRET}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—		—	—	—	—	—	—	1	—	
1	— (1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	$DEVENT_{DQTAG}$, DDAM

1. IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8 and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$=1. Only software-owned bits or fields as shown in *Table 966* are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

57.3.4 Debug Event Select register (DEVENT) register

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Core (e200z215An3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in *Figure 1014*.

DQTAG	0	DEVNT

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

SPR 975; Reset(1): 0x0

Figure 1014. Debug Event Select register (DEVENT) register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b.** Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

Table 967 provides bit definitions for the Debug Event Register.

Table 967. DEVENT field descriptions

Bit	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 No signal is asserted *xxxxxxx*1 **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock *xxxxxx*1*x* **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock *xxxxx*1*xx* **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock *xxxx*1*xxx* **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock *xxx*1*xxxx* **p_devnt_out[4]** is asserted for one clock *xx*1*xxxxx* **p_devnt_out[5]** is asserted for one clock *x*1*xxxxxx* **p_devnt_out[6]** is asserted for one clock 1*xxxxxxx* **p_devnt_out[7]** is asserted for one clock

57.3.5 Debug Data Acquisition Message (DDAM) register

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. See the "Data Acquisition Messaging" section in the Core (e200z215An3) Nexus 3 Module chapter for details.

The DDAM register is shown in *Figure 1015*.

DDAM																															
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
SPR 576; Reset(1): 0x0																															

Figure 1015. Debug Data Acquisition Message (DDAM) register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 968 provides bit definitions for the Debug Data Acquisition Message Register.

Table 968. DDAM field descriptions

Bit	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

57.4 Using debug resources for stack limit checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access that generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in *Table 959*. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to '1000' or the $DBCR8_{DAC3CFG}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '10', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '01', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a

single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in *Table 959* and *Table 962* in *Section 57.3.2, Debug Control and Status registers*. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

57.5 External debug support

External debug support is supplied through the OnCE controller serial interface, which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of *Section 57.2, Software debug events and exceptions* refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note: On the initial setting of $EDBCR0_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information that could cause unintended operation.

*Note: It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.*

*Note: $EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all hardware debug events are blocked. Watchpoints are not blocked.*

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The

OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

57.5.1 External debug registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug mode.

57.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in *Figure 1016*.

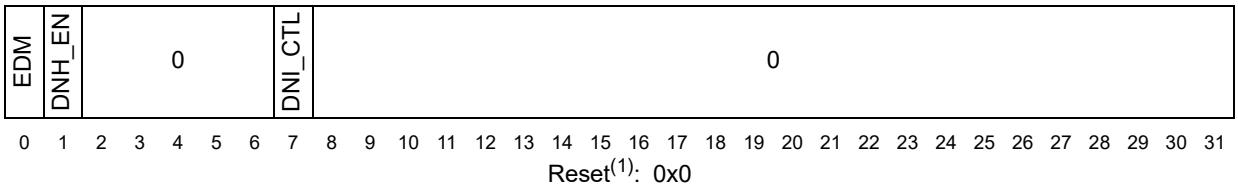


Figure 1016. External Debug Control Register 0 (EDBCR0)

1. EDBCR0 is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**.

Table 969 provides bit definitions for External Debug Control Register 0.

Table 969. EDBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0.
1	DNH_EN	**dnh** Instruction Enable 0 Execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 Execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	—	Reserved
7	DNI_CTL	**dni** Instruction Control 0 When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSR_{DE}=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. **Note:** This control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1.
8:31	—	Reserved

57.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in *Figure 1017*.

Figure 1017. External Debug Status Register 0 (EDBSR0)

IDE	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	0	CIRPT	CRET	VLES	DAC_OFST	0
0	1	2	3	4	5	6	7	8	9 10 11	12	13	14	15	16	17 18 19 20	21	22	23	24	25	26	27	28 29 30 31	

Read/Write; Reset(1) - 0x0000_0000

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCR0_{EDM}$=0.

Table 970 provides bit definitions for External Debug Status Register 0.

Table 970. EDBSR0 field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store that is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events that are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$
24	—	Reserved

Table 970. EDBSR0 field descriptions(Continued)

Bit	Name	Description
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3‘b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3‘b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

57.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A ‘1’ stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events that are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in *Figure 1018*.

Figure 1018. External Debug Status Register Mask 0 (EDBSRMSK0)

0	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	0	CIRPT	CRET	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Read/Write; Reset(1) - 0x0000_0000

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCR0_{EDM}$=0.

Table 971 provides bit definitions for External Debug Status Register Mask 0.

Table 971. EDBSRMSK0 field descriptions

Bit	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$

Table 971. EDBSRMSK0 field descriptions (Continued)

Bit	Name	Description
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

57.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events. EDDEAR is update by hardware with the effective address of the load or store operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits that are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in *Figure 1019*.

Data Effective Address

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

Read/Write; Reset: unaffected

Figure 1019. External Debug Data Effective Address Register (EDDEAR)

57.5.2 OnCE introduction

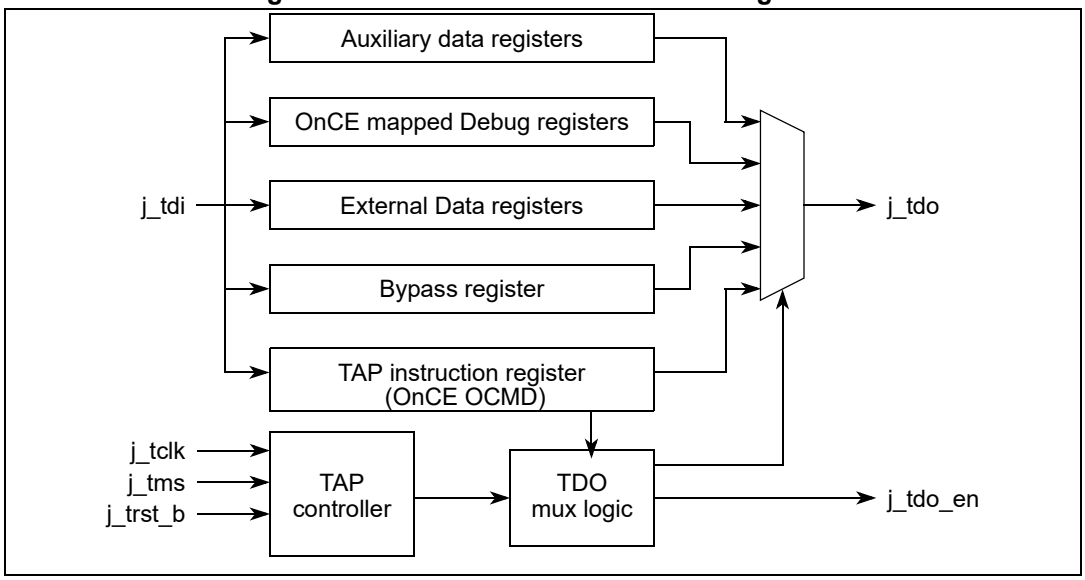
The e200z215An3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z215An3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

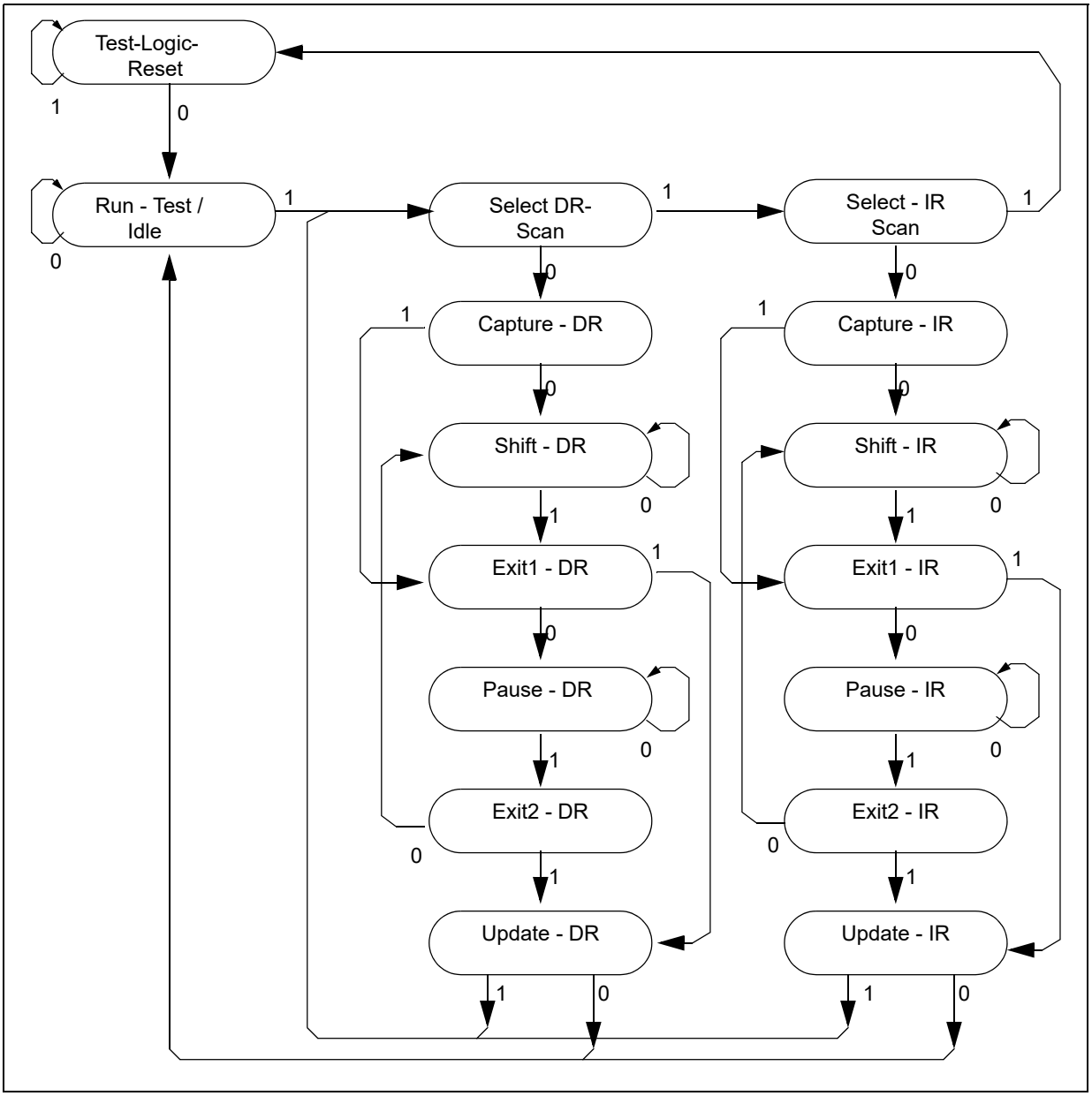
The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

Figure 1020. 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 1021. OnCE controller implementation



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core, which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), *Section 57.5.6.1, 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 will

indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See *Section 57.5.6.2, OnCE Command (OCMD) register*). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

57.5.3 JTAG/OnCE pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

Table 972 details the primary JTAG/OnCE interface signals.

Table 972. JTAG/OnCE primary interface signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en(1)	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z215An3) Core Complex Overview.

57.5.4 OnCE internal interface signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

57.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

57.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

57.5.4.3 CPU address and attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

57.5.4.4 CPU data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

57.5.5 OnCE interface signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks that may need information pertaining to debug operation.

57.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE commands are executed by the OnCE unit, and all other commands default to a “Bypass” command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

57.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z215An3 interface) provides a fast means of entering the Debug mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug mode. Once Debug mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems that 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.

57.5.5.3 OnCE Debug output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control that are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either “No Register Selected” or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See *Section 57.5.6.2, OnCE Command (OCMD) register* for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

57.5.5.4 CPU Clock On input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU’s **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z215An3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

57.5.5.5 Watchpoint events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, and the Performance Monitor control register settings. Refer to *Table 979* for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

57.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

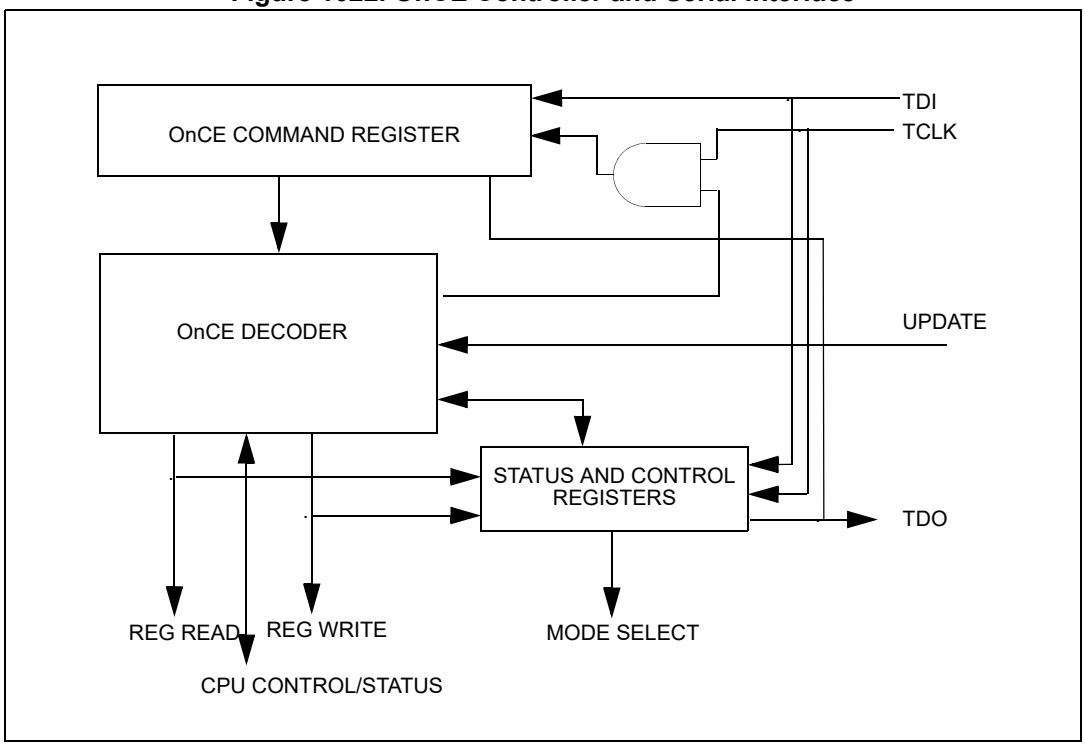
This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates “no register selected”. This signal will assert regardless of whether the CPU is currently in debug mode. It may be monitored by external logic to cause a synchronous exit from debug mode of other modules

(but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with “no register selected”, and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs that are not in debug mode will ignore the command.

57.5.6 OnCE controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. *Figure 1022* is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR, and Update-DR states.

Figure 1022. OnCE Controller and Serial Interface



57.5.6.1 OnCE Status register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Figure 1023. OnCE Status register

Table 973 provides bit definitions for the OnCE Status register.

Table 973. OnCE Status register field descriptions

Bit	Name	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 will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
2	—	Reserved, set to zero
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

57.5.6.2 OnCE Command (OCMD) register

The OnCE Command (OCMD) register is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in *Figure 1024*. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

R/W	GO	EX	RS[0:6]
0	1	2	3 4 5 6 7 8 9

Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por**, or while in the Test_Logic_Reset state

Figure 1024. OnCE Command (OCMD) register

Table 974 provides bit definitions for the OCMD.

Table 974. OCMD field descriptions

Bit	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo** during the first 32 clocks of Shift-DR. 0 Write the data associated with the command into the register specified by RS[0:6] 1 Read the data contained in the register specified by RS[0:6]
1	GO	Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction 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 will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. **Note:** Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See *Section 57.5.9.6, Exiting Debug mode and interrupt blocking*. 0 Inactive (no action taken) 1 Execute instruction in IR

Table 974. OCMD field descriptions (Continued)

Bit	Name	Description
2	EX	Exit Command Bit 0 Remain in debug mode 1 Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to “No Register Selected”. Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. **Note:** If the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. **Note:** Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See *Section 57.5.9.6, Exiting Debug mode and interrupt blocking*.
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. *Table 975* indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 975 indicates the OnCE register addresses.

Table 975. OnCE register addressing

RS[0:6]	Register Selected
000 0000	Reserved
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)

Table 975. OnCE register addressing(Continued)

RS[0:6]	Register Selected
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	Data Value Compare 1 (DVC1) — *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) — *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Reserved for Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select

Table 975. OnCE register addressing(Continued)

RS[0:6]	Register Selected
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Reserved for Cache Debug Access Control Register (CDACNTL)
111 1011	Reserved for Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3 Access (see *Chapter 66: e200z215An3 Nexus 3 Module*)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE(1)
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The OnCE Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

Single-stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (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 will correspond to the first instruction of the exception handler.

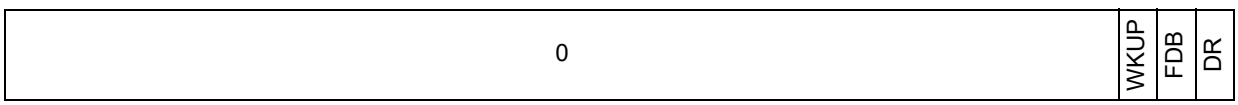
Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of $EDBCR0_{EDM}$. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

57.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z215An3 core into debug mode and to enable / disable sections of the OnCE control logic. The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in *Figure 1025*.

Figure 1025. OnCE Control Register (OCR)



Reset - 0xo000_0000 on **m_por**, **j_trst_b**, or entering Test_logic_Reset state

Table 976 provides bit definitions for the OnCE Control Register.

Table 976. OCR field descriptions

Bit	Name	Description
0:28	—	Reserved
29	WKUP	Wakeup Request Bit This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.
30	FDB	Force Breakpoint Debug Mode Bit This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo-instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. **Note:** This bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter Debug mode. The CPU will indicate that Debug mode has been entered via the data scanned out in the shift-IR state. 0 No Debug mode request 1 Unconditional Debug mode request When the DR bit is set, the processor will enter Debug mode at the next instruction boundary.

57.5.7 Access to debug resources

Resources contained in the OnCE Module that do not require the processor core to be halted for access may be accessed while the e200z215An3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z215An3 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 e200z215An3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCR_{WKUP} control bit.

In addition, since 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 977 provides a list of access requirements for OnCE registers.

Table 977. OnCE register access requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 $_{EDM}$ = 1 for write access	Requires m_clk active for write access	Requires CPU to be halted for read access	Requires CPU to be halted for write access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*(1)	
DBCR0	Y	Y	Y	N	*(1)	*EDBCR0$_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*(1)	
DEVENT	Y	Y	Y	N	*(1)	
EDBRAC0 (DBERC0)	Y	N	Y	N	*(1)	
DBSR	Y	Y	Y	N(2)	*(1)	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*(1)	
JTAG ID	N	N	—	N	—	Read-only
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	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. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

57.5.8 Methods of entering Debug mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See *Section 57.5.5.1, OnCE Enable (jd_en_once)*.

57.5.8.1 External debug request during RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b**, and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z215An3 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 e200z215An3 core does not execute an instruction before entering Debug mode, although the first instruction to be executed will be fetched prior to entering Debug mode in order to properly initialize the CPUSCR.

57.5.8.2 Debug request during RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

57.5.8.3 Debug request during normal activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

57.5.8.4 Debug request during Waiting, Halted, or Stopped state

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled along with other status bits from the **j_tdo** output during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

57.5.8.5 Software request during normal activity

Upon executing a '**bkpt'** pseudo-instruction (for e200z215An3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit

is true), and $EDBCR0_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt'** pseudo-instruction has entered the instruction register.

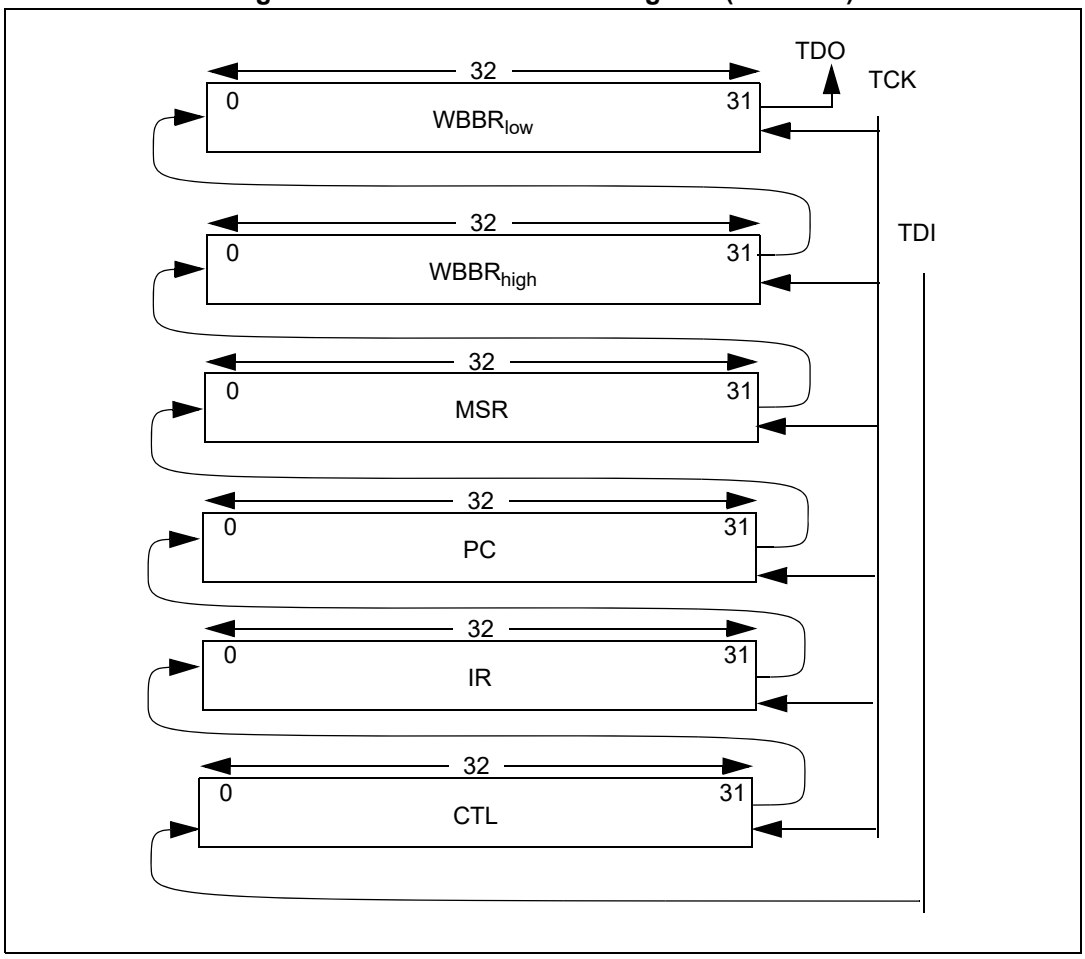
57.5.8.6 Debug notify halt instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z215An3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

57.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. *Figure 1026* 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 1026. CPU Scan Chain Register (CPUSCR)



57.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

57.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the

external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

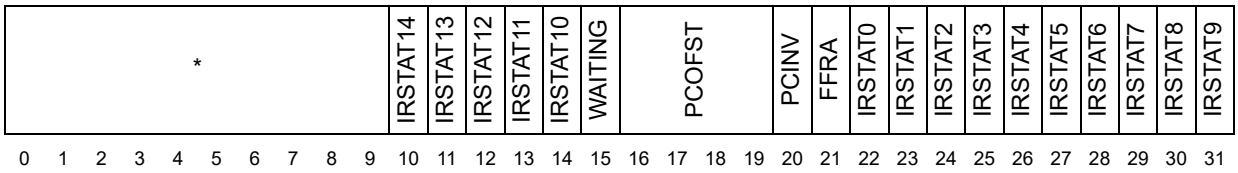


Figure 1027. Control State Register (CTL)

Table 978. CTL field descriptions

Bit	Name	Description
0:9	*	Internal State Bits These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See *Section 57.5.5.3, OnCE Debug output (jd_debug_b)*, which is not part of the normal program execution flow, these bits should be set to a 0.
10	IRSTAT14	IR Status Bit 14 This control bit is reserved.
11	IRSTAT13	IR Status Bit 13 This control bit indicates an Instruction Address Compare 8 event status for the IR. 0 No Instruction Address Compare 8 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 8 event occurred on the fetch of this instruction.
12	IRSTAT12	IR Status Bit 12 This control bit indicates an Instruction Address Compare 7 event status for the IR. 0 No Instruction Address Compare 7 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 7 event occurred on the fetch of this instruction.
13	IRSTAT11	IR Status Bit 11 This control bit indicates an Instruction Address Compare 6 event status for the IR. 0 No Instruction Address Compare 6 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 6 event occurred on the fetch of this instruction.
14	IRSTAT10	IR Status Bit 10 This control bit indicates an Instruction Address Compare 5 event status for the IR. 0 No Instruction Address Compare 5 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 5 event occurred on the fetch of this instruction.
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 will determine whether the CPU re-enters the waiting state on a go+exit. 0 CPU was not in the waiting state when debug mode was entered 1 CPU was in the waiting state when debug mode was entered

Table 978. CTL field descriptions (Continued)

Bit	Name	Description
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
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 will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered. 0 No error condition exists. 1 Error condition exists. PC and IR are corrupted.
21	FFRA	Feed Forward RA Operand Bit This control bit causes the content of the WBBRlo to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.For most LSP instructions using rA||rB as a 64-bit source operand, WBBRhi, lo is used to supply the 64-bit source value. This allows the debug firmware to update processor registers — initialize the WBBRlo 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 $WBBR_{lo}$ used as operand value. **Note:** Not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent.
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 is reserved.

Table 978. CTL field descriptions (Continued)

Bit	Name	Description
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 event 1 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 1 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 event 3 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 3 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. 0 No Parity Error occurred on the fetch of this instruction. 1 Parity Error occurred on the fetch of this instruction.
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 A Precise External Termination Error occurred on the fetch of this instruction.
30	IRSTAT8	IR Status Bit 8 This control bit indicates the PowerISA VLE status for the IR. 0 IR contains a BookE instruction. (unused encoding) 1 IR contains a PowerISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit. **Note:** This bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).
31	IRSTAT9	IR Status Bit 9 This control bit is reserved.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be written with the FFRA bit set as appropriate, and all other bit set to ‘0’, and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit that was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU,

the PC value must be backed up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC + 4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events that may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be reentered prior to any further instruction execution.

57.5.9.3 Program Counter (PC) register

The PC is a 32-bit register that stores the value of the program counter that was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

57.5.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed, which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to *Section 57.5.9.2, Control State Register (CTL)* for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions that do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

57.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

57.5.9.6 Exiting Debug mode and interrupt blocking

When exiting debug mode with a Go+Exit, “asynchronous” interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will reenter debug mode without executing an instruction following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

57.5.10 Reserved registers (Reserved)

The reserved registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

57.6 Watchpoint support

e200z215An3 supports the generation and signalling of watchpoints when operating in internal debug mode ($DBCR0_{IDM}$=1) or in external debug mode ($EDBCR0_{EDM}$=1). Watchpoints are indicated with a dedicated set of interface signals. The **jd_watchpt[0:31]** output signals are used to indicate that a watchpoint has occurred. Certain watchpoints however (DEVENT-based, DTC-based, and Performance Monitor watchpoints) are not qualified with $EDBCR0_{EDM}$ or $DBCR0_{IDM}$.

Each debug address compare function (IAC1–8, DAC1–4), as well as other event types are capable of triggering a watchpoint output. The DBCRx control fields are used to configure watchpoints, regardless of whether events are enabled in DBCR0. Watchpoints may occur whenever an associated event would have been posted in the Debug Status Register if enabled. No explicit enable bits are provided for watchpoints; they are always enabled by definition, although the data address compare watchpoints may be controlled for read and write accesses via configuration fields in DBCR4, 7, and 9. During a debug session, debug events (other than PMI, DEVT1 and DEVT2) with a corresponding DBSR bit are blocked from asserting a watchpoint. The Performance Monitor, DEVNT-based and DTC-based watchpoints are not blocked during a debug session. If not desired, for address-based events the base address values for these events may be programmed to an unused system address. MSR_{DE} has no effect on watchpoint generation.

External logic may monitor the assertion of these signals for debugging or triggering purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus3 module also monitors assertion of a portion of these output signals (**jd_watchpt[0:31])** for various development control purposes. (See the “Watchpoint Trace Messaging” section in the Core (e200z215An3) Nexus 3 Module chapter.)

Note that these signals are **m_clk** domain signals, not **j_tclk** domain signals.

Table 979 shows the **jd_watchpt[0:31]** output signal assignments to various debug events.

Table 979. Watchpoint output signal assignments

Signal Name	Type	Description
jd_watchpt[0]	IAC1	Instruction Address Compare 1 watchpoint Asserted whenever an IAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[1]	IAC2	Instruction Address Compare 2 watchpoint Asserted whenever an IAC2 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[2]	IAC3	Instruction Address Compare 3 watchpoint Asserted whenever an IAC3 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[3]	IAC4	Instruction Address Compare 4 watchpoint Asserted whenever an IAC4 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[4]	DAC1	Data Address Compare 1 watchpoint Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC1CFG}$ settings
jd_watchpt[5]	DAC2	Data Address Compare 2 watchpoint Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC2CFG}$ settings
jd_watchpt[6]	DAC3	Data Address Compare 3 watchpoint Asserted whenever a DAC3 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC3CFG}$ settings
jd_watchpt[7]	DAC4	Data Address Compare 4 watchpoint Asserted whenever a DAC4 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC4CFG}$ settings
jd_watchpt[8]	IAC5	Instruction Address Compare 5 watchpoint Asserted whenever an IAC5 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[9]	IAC6	Instruction Address Compare 6 watchpoint Asserted whenever an IAC6 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[10]	DEVT1	Debug Event Input 1 watchpoint Asserted whenever a DEVT1 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[11]	DEVT2	Debug Event Input 2 watchpoint Asserted whenever a DEVT2 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[12]	DEVNT0	Debug Event Output 0 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[12]
jd_watchpt[13]	DEVNT1	Debug Event Output 1 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[13]

Table 979. Watchpoint output signal assignments(Continued)

Signal Name	Type	Description
jd_watchpt[14]	IAC7	Instruction Address Compare 7 watchpoint Asserted whenever an IAC7 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[15]	IAC8	Instruction Address Compare 8 watchpoint Asserted whenever an IAC8 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[16]	IRPT	Interrupt watchpoint Asserted whenever an IRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[17]	RET	Return watchpoint Asserted whenever a RET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[18]	CIRPT	Critical Interrupt watchpoint Asserted whenever a CIRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[19]	CRET	Critical Return watchpoint Asserted whenever a CRET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[20]	DEVNT2	Debug Event Output 2 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[20]
jd_watchpt[21]	DEVNT3	Debug Event Output 3 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[21]
jd_watchpt[22]	PMEVENT	Performance Monitor Event input watchpoint Asserted whenever **p_pm_event** transitions from a ‘0’ to a ‘1’ while **m_clk** is running
jd_watchpt[23]	PMC0	Performance Monitor Counter 0 watchpoint Asserted whenever PMC0 triggers an event based on $PMLCa0_{PMP}$
jd_watchpt[24]	PMC1	Performance Monitor Counter 1 watchpoint Asserted whenever PMC1 triggers an event based on $PMLCa1_{PMP}$
jd_watchpt[25]	PMC2	Performance Monitor Counter 2 watchpoint Asserted whenever PMC2 triggers an event based on $PMLCa2_{PMP}$
jd_watchpt[26]	PMC3	Performance Monitor Counter 3 watchpoint Asserted whenever PMC3 triggers an event based on $PMLCa3_{PMP}$
jd_watchpt[27]	—	Reserved
jd_watchpt[28]	TRAP	TRAP watchpoint Asserted whenever an TRAP debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[29]	DTC1	Data Trace Control Range 1 watchpoint Asserted whenever an access meets the conditions for DTC Range 1

Table 979. Watchpoint output signal assignments(Continued)

Signal Name	Type	Description
jd_watchpt[30]	DTC2	Data Trace Control Range 2 watchpoint Asserted whenever an access meets the conditions for DTC Range 2
jd_watchpt[31]	DTC3	Data Trace Control Range 3 watchpoint Asserted whenever an access meets the conditions for DTC Range 3

57.7 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:

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation.
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to ‘1’. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set.
- Clear the DBSR status bits.
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set.

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 57.5.9.2, 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 reentered the Debug state and that no ERR condition occurred.

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set.
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 57.5.9.2, 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.
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit.
- The debugger should write the DBSR to a cleared state.
- The debugger should 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 57.5.9.2, 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.

58 Debug and Calibration Interface (DCI)

58.1 Introduction

The Debug and Calibration Interface (DCI) module provides debug and calibration features for the MCU. It includes a standard 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool using a low frequency clock interface.

It also features a software based debug mode which can be controlled by the MCU without using an external tool. Additionally, the DCI provides features for debug control using break-points and synchronous restart of the cores. This chapter describes the DCI features for the Production device (PD) only.

58.1.1 Features

The DCI features are the following:

- Debug mode enable control for connected tool or software
- 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for CPU when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug reset control

58.1.2 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG—JTAG signals coming from the Production device JTAG pads. This is the standard 5-pin JTAG interface and is the default operating mode.
- M.JTAG—JTAG signals generated by JTAGM module using software debug mode.
  - In software debug mode, there is no tool, and debug software running on the MCU generates the JTAG packets and writes them to JTAGM using special debug registers in the JTAGM module.

By default the DCI operates in standard 5-pin JTAG mode (IEEE 1149.1). It can be configured in 3-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 5-pin JTAG mode.

When reduced pin JTAG mode (IEEE 1149.7) is configured TMSC pin is configured in bidirectional mode during operation.

The DCI also provides a centralized break control for system IPs and cores, which can be controlled by an external tool as well as the internal debug peripherals such as GTM/SPU, etc. The DCI also provides the Event Out ($\overline{\text{EVTO}}$) signals to the debug tool in case of the occurrence of any internal debug event.

Figure 1028 shows the DCI block diagram.

Figure 1028. DCI block diagram

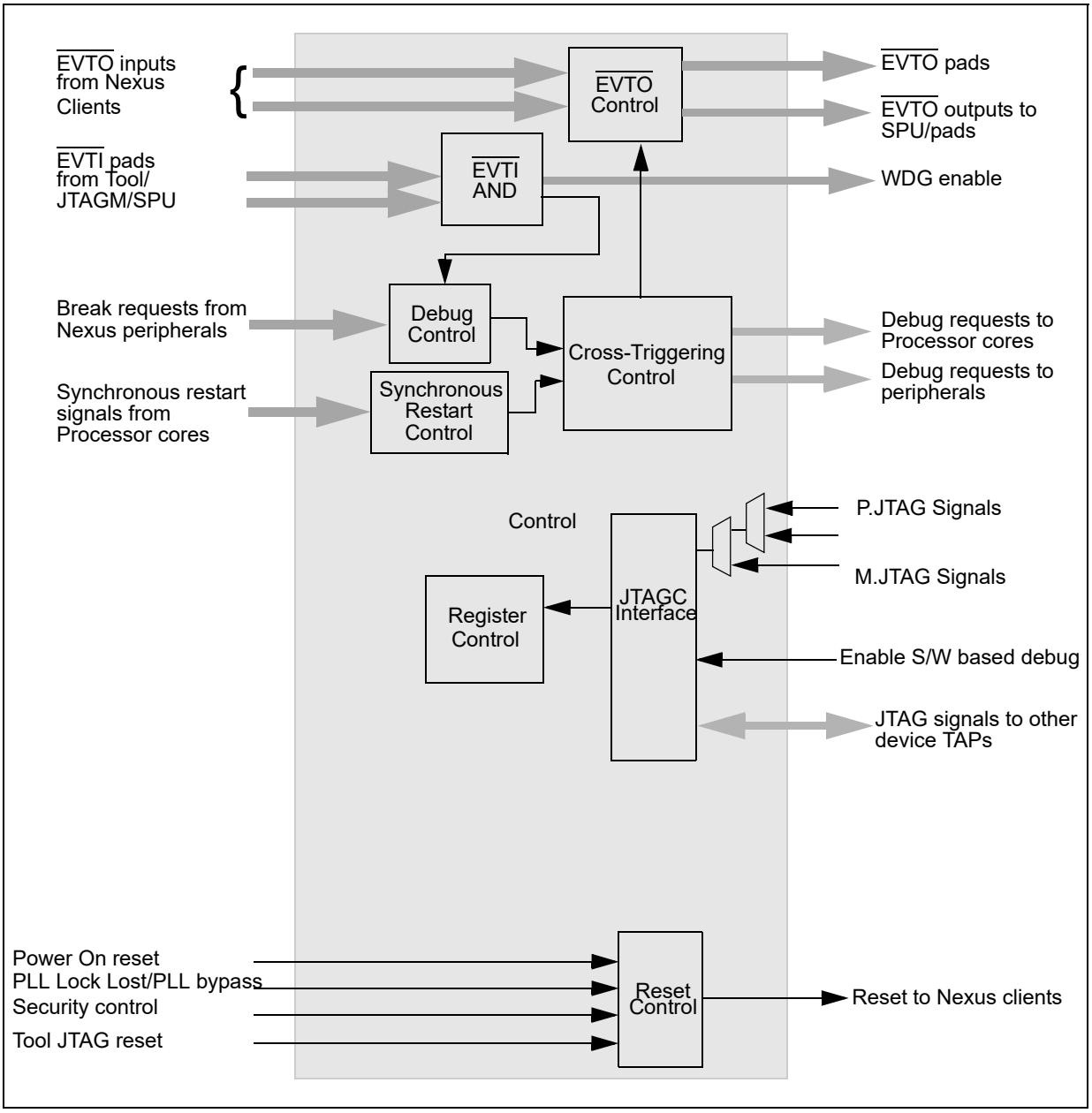
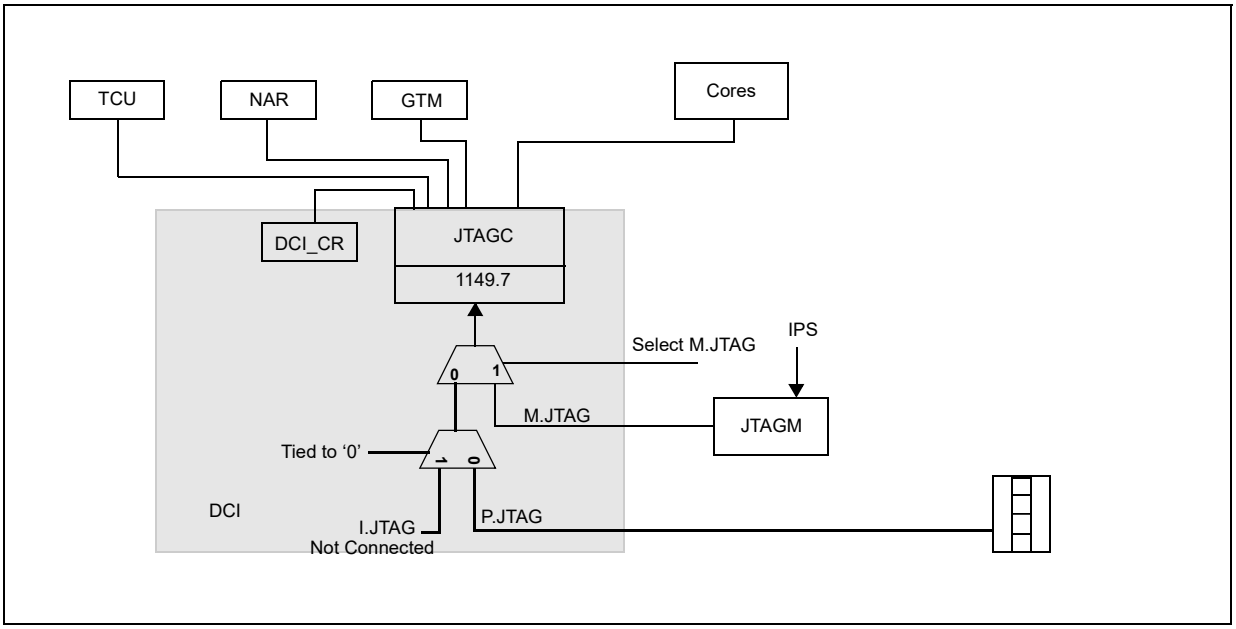


Figure 1029 shows the DCI configuration for the standard production package.

Figure 1029. DCI in a standard configuration



58.1.3 Operating modes

Selection between various JTAG sources is done using a JTAGM register bit, JTAGM_MCR[DTM]. The JTAGM bit, JTAGM_MCR[DTM], is programmable via memory-mapped registers. JTAGM is selected as a JTAG source when JTAGM_MCR[DTM] is set.

Table 980 shows a summary of the configuration options.

Table 980. DCI JTAG source selection

JTAGM_MCR[DTM]	DCI Source	JCOMP
0	P.JTAG (Standard JTAG)	Asserted
1	M.JTAG (Software)	Negated

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/Core supplies
- JCOMP reset

The DCI puts the JTAGC TAP in reset when either power-on/low-voltage-detect reset is asserted or JCOMP is negated.

The M.JTAG provides an alternate interface to the debug logic.

When no debug tool is connected, device power-on reset puts the DCI in standard 5-pin JTAG mode. This configures the TDO pin in output mode with the reset value (1'b0). There is no change in port values if no debug tool is connected.

Table 981 contains the operating mode descriptions.

Table 981. 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.

58.1.3.1 Software enabled debug and calibration mode

Apart from the JTAG sources, such as a JTAG-based tool , it is possible to generate JTAG sequences through software programming using JTAGM IP.

To support a software debug session via a CAN or SPI link, the software can enable debug and calibration mode via an memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to the JTAGM documentation. When enabled, the DCI is configured to receive M.JTAG signals which are driven by JTAGM registers accessible through software. Software can control all DCI features like any other JTAG source.

After a system reset, the JTAGM_MCR[DTM] bit is reset and the JCOMP pad is pulled-low, thus keeping the DCI in the reset state. Therefore, software is required to first change to a non-secured mode. Then it can program the JTAGM_MCR[DTM] bit, to select M.JTAG as the source inside the DCI, and then set the JTAGM_MCR[jtagm_JCOMP] bit present in the JTAGM, which takes all debug modules out of reset.

There is a software programmable bit, MCR[jtagm_JCOMP], present in JTAGM block which, when programmed to 0, asserts reset for the JTAGC block, thus putting the debug and calibration logic in reset state. This bit is used during software based debug to initialize the debug and calibration logic.

Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

Note: It is recommended to utilize either hardware based debug (JTAG) or software based debug. If only one of the debug method is used, all three connection methods (connect and reset target, connect and break application, connect and keep application running) are supported.

Note: If both debug methods are used simultaneously, then an external tool may arbitrarily connect when the SoC has already been halted using the software debug mode. In this case, two consecutive assertions of external JCOMP are necessary to allow the external tool to take control of debug session.

58.1.3.2 Nexus reset control

The JCOMP input that is used as the primary reset signal for the DCI is also used by the DCI to generate a single-bit reset signal for other Nexus blocks. It is used to reset all debug

functions without affecting any system level functions. The DCI provides the reset signal for all Nexus blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect
- JCOMP
- SoC Debug_Enable signal
- System PLL is out of lock state and not bypassed

Asserting power-on reset, negating JCOMP, or negating the Debug_Enable signal, results in asynchronous entry into reset state. Following negation of power-on reset, the Nexus reset signal remains asserted until the system clock achieves lock or in bypass mode, as indicated by the system clock lock status signal, and Debug_Enable is asserted and JCOMP is asserted. Any subsequent loss of PLL lock does not generate a Nexus reset. This reset signal is unaffected by other sources of reset.If Debug_Enable is de-asserted by the SoC security modules, then the DCI holds the debug and trace logic in reset. The Debug_Enable signal is generated by security logic after functional or destructive reset based on the device censored mode, the PASS_LOCK3[DBL] control, and the device life cycle. If the Debug_Enable signal is asserted by the security logic, it is possible to trace through functional resets

The following table shows how the Nexus reset is asserted/negated based on various signals.

Table 982. Nexus reset truth table

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	X	Assert
X	Negate	X	X	Assert
X	X	Disable	X	Assert
X	X	X	Unlocked	Assert
Negate	Assert	Enable	Locked	Negate

58.1.3.3 System watchdog control

The DCI is also used to enable or disable the Software Watchdog Timer (SWT). The DCI provides a single bit SWT enable signal which is used to control all SWTs in the MCU. For this, the $\overline{\text{EVTI}}$[0] pin is used. If the $\overline{\text{EVTI}}$[0] pin remains high when the tool asserts JCOMP, then the SWT is enabled. But if the tool drives the $\overline{\text{EVTI}}$[0] pin low before asserting JCOMP pin, then the SWT is disabled. By default the SWT is enabled and a tool has an option to enable/disable the SWT using the $\overline{\text{EVTI}}$[0] pin.

58.1.3.4 Break control

The DCI provides central break control among the CPUs and non-CPU debug modules such as GTM, DMA, Watchdog and other peripherals.

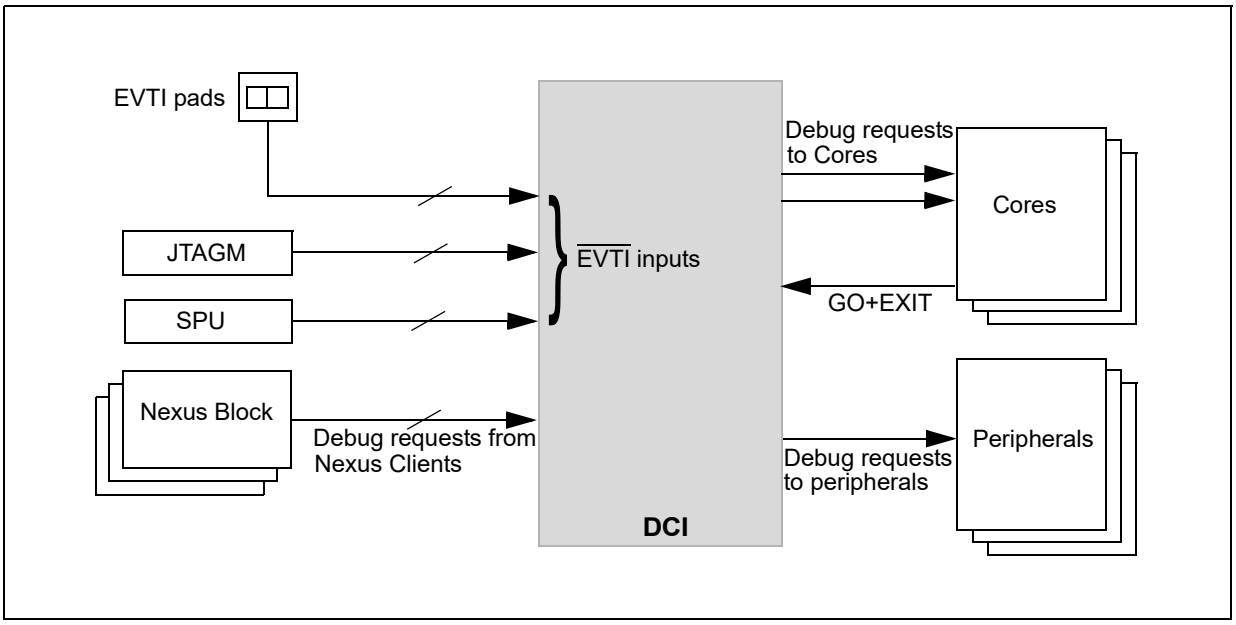
There are various different input sources that provide debug requests:

- $\overline{\text{EVTI}}$[1:0] signals from EVTI pads
- $\overline{\text{EVTI}}$[1:0] signals from JTAGM
- $\overline{\text{EVTI}}$[1:0] signals from the Sequence Processing Unit (SPU)

The DCI generates following debug request signals:

- Debug Request signal to be used for all non-CPU peripherals
- External Debug Request to be used to request external debug event to processor cores
- OnCE Debug Request to be used to request OnCE debug event to processor cores
- All break control requests are issued simultaneously to CPUs and peripherals, then synchronized to each clock domain

Figure 1030. DCI debug control mechanism



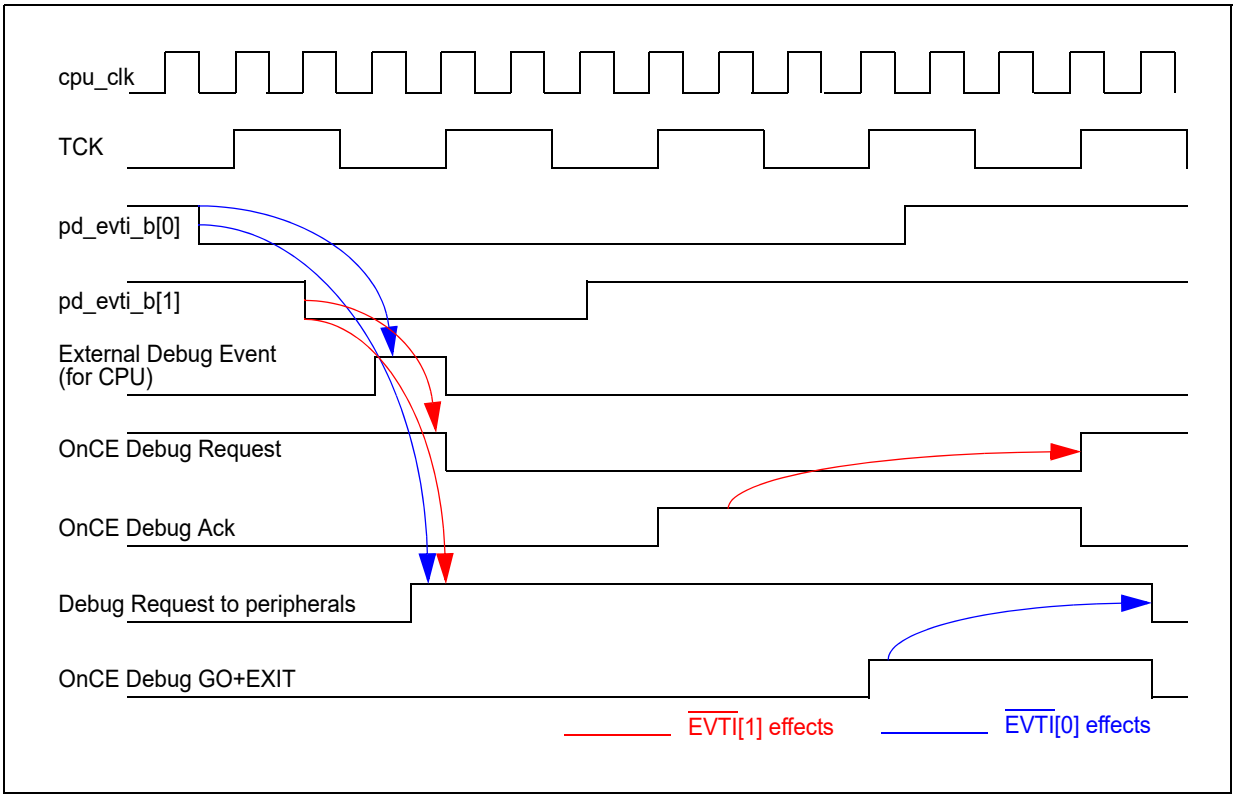
The various sources of $\overline{\text{EVTI}}$[1:0] inputs are synchronized and logically ANDed in the DCI to generate two debug requests DCI_$\overline{\text{EVTI}}$[1:0] for the processor cores and non-CPU peripherals.Of these two inputs, $\overline{\text{EVTI}}$[0] is used to generate debug requests 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 $\overline{\text{EVTI}}$[1:0] signals coming from the pads are enabled for debug control using DCI_CR[5:4] bits.

In order to selectively enable/disable debug requests to system peripherals, the respective debug enable bit inside the peripheral should be used.

Figure 1031 shows the timing sequence.

Figure 1031. DCI debug signal generation



58.1.3.5 Cross-triggering control

Many of the blocks in the device, including the Nexus trace blocks and more sophisticated peripherals, have debug features that allow them to request device-wide breaks to enter debug mode. Many blocks can be programmed to halt operation upon debug mode entry or ignore it and continue normal operation. All debug mode entry requests are input to the DCI, ORed together and the result is used to produce the debug requests to the CPUs and non-CPU peripherals.

The cross-triggering feature is implemented through selectively enabling debug control bits in the respective modules. Processor cores (enabled using DBCR0[DEVT1] bit) and non-core peripherals can be programmed to halt operation through their respective control bits. Cross-triggering provides a selective feature to put modules in debug mode upon receiving halt request signals from Nexus modules.

58.1.3.6 Synchronous restart control

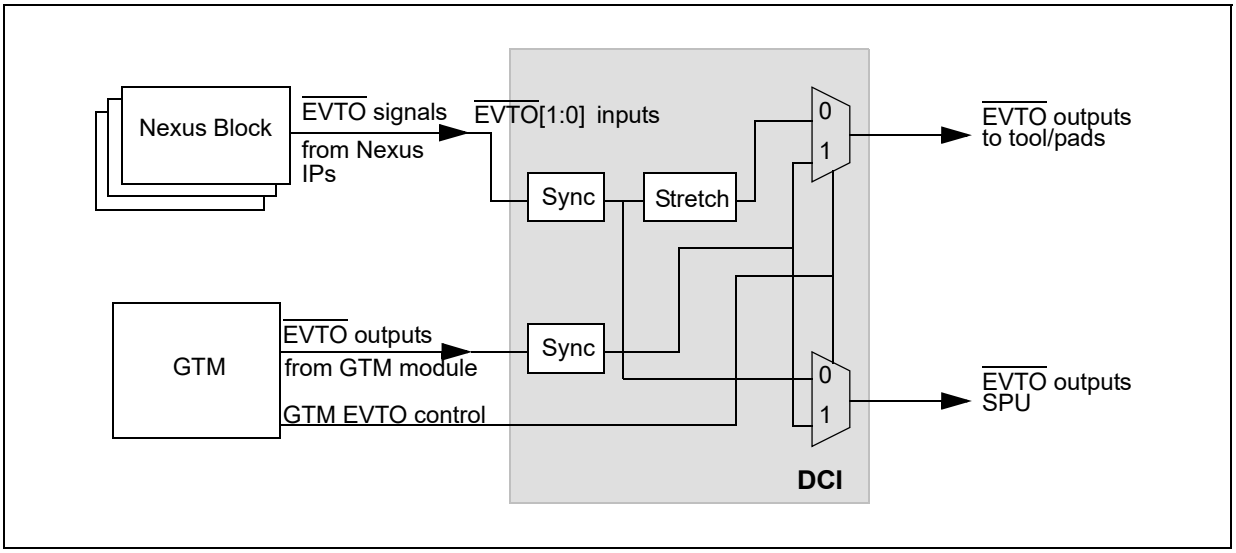
The DCI implements a synchronous restart control feature to synchronously restart all debug peripherals (CPU and non-CPU peripherals which are put in debug state). This is accomplished with the following steps:

1. The tool sequentially scans into CPUSCR for each processor core to restore the proper processor state.
2. The tool scans into the Production device tap controller the AUX_PARALLEL instruction, which then selects all processor core tap controllers in parallel.
3. To synchronously exit any combination of processor cores from debug mode, the tool simultaneously scans into all processor core tap instruction registers (OCMD) the 10-bit sequence 0x180 (i.e. 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.

58.1.3.7 EVTO management

The EVTO management block diagram is presented in *Figure 1032*.

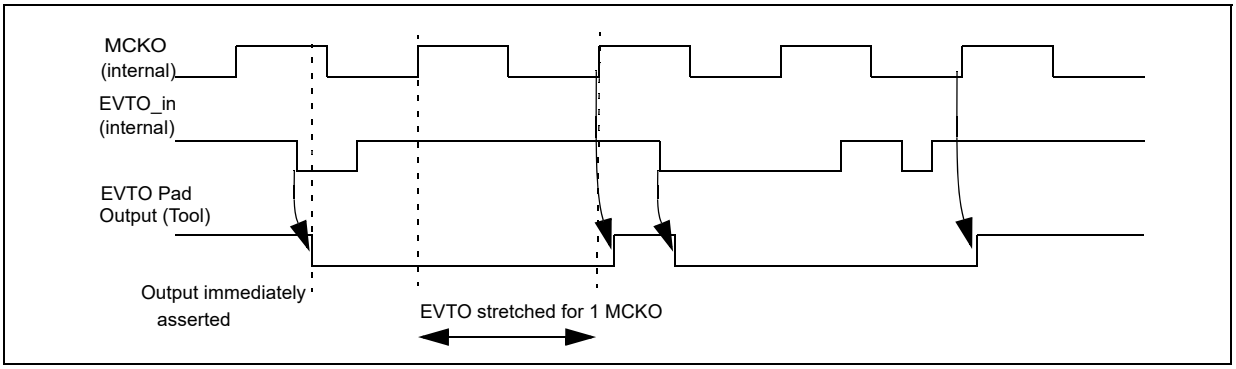
Figure 1032. DCI EVTO management



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 1033*). 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.

The DCI also allows the GTM to take control of the EVTO outputs by controlling the EOS0/EOS1 fields of the GTMDI_DC register. If these fields are programmed to HALT/TIM/TOM/ATOMEVTO value, the GTM EVTO outputs are directly available on the ports bypassing the stretch logic.

Figure 1033. DCI EVTO timing diagram



58.1.3.8 Reset control

The DCI includes control bits to allow a connected tool to force a reset of the connected device. The description is as follows:

- Force functional reset—This reset is generated by programming the DCI_CR[FPD_FUNC] bit. When set, it forces functional reset. This bit has to be reset by the tool to negate the reset signal.
- Force POR or destructive reset—This reset is generated by programming DCI_CR[FPD_DEST] bit. When set it forces destructive reset. This bit has to be reset by the tool to negate the reset signal.

These reset signals are directly connected to the respective DCI_CR bits. The DCI does not reset the DCI_CR bits after generation of the respective reset signals. The tool must specifically reset the appropriate DCI_CR bits to remove the reset signals. These reset control bits can be set/reset through JTAG programming, and only reset through JCOMP assertion or Power-on-reset.

58.2 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of five signals (TMS/TDI/TDO/TCK/JCOMP) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

58.3 Register description

The PD DCI module contains two registers, DCI_CR and DCI_PINCR, which are present inside the JTAGC module. These registers are only controlled using the JTAGC TAP and no memory-mapped registers are present.

58.3.1 DCI control register (DCI_CR)

The DCI_CR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. *Figure 1034* shows the format of the DCI_CR.

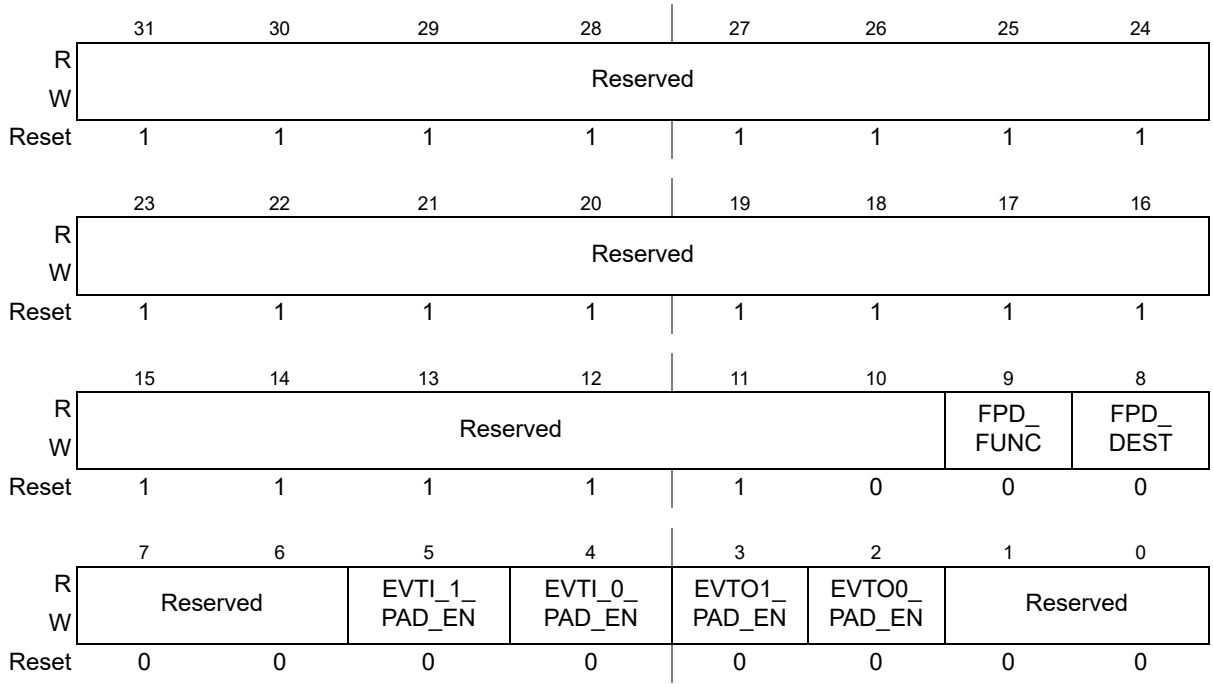


Figure 1034. DCI control register (DCI_CR)

The DCI_CR is described in *Table 983*.

Table 983. DCI_CR field descriptions

Field	Description
9 FPD_FUNC	Force Functional reset 0 Do not force functional reset 1 Force functional reset
8 FPD_DEST	Force Destructive reset 0 Do not force destructive reset 1 Force destructive reset
5 EVTI1_PAD_EN	$\overline{\text{EVTI}}$[1] feature enable 0 $\overline{\text{EVTI}}$[1] disabled (default) 1 $\overline{\text{EVTI}}$[1] enabled
4 EVTI0_PAD_EN	$\overline{\text{EVTI}}$[0] feature enable 0 $\overline{\text{EVTI}}$[0] disabled (default) 1 $\overline{\text{EVTI}}$[0] enabled
3 EVTO1_PAD_EN	Output enable for $\overline{\text{EVTO}}$[1] pad 0 $\overline{\text{EVTO}}$[1] pad output not enabled (default) 1 $\overline{\text{EVTO}}$[1] pad output enabled
2 EVTO0_PAD_EN	Output enable for $\overline{\text{EVTO}}$[0] pad 0 $\overline{\text{EVTO}}$[0] pad output not enabled (default) 1 $\overline{\text{EVTO}}$[0] pad output enabled

58.3.2 DCI EVTx 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. For a description of the DCI_PINCR functionality, see the Calibration and Debug configuration chapter which describes how the modules are configured.

59 JTAG Controller (JTAGC)

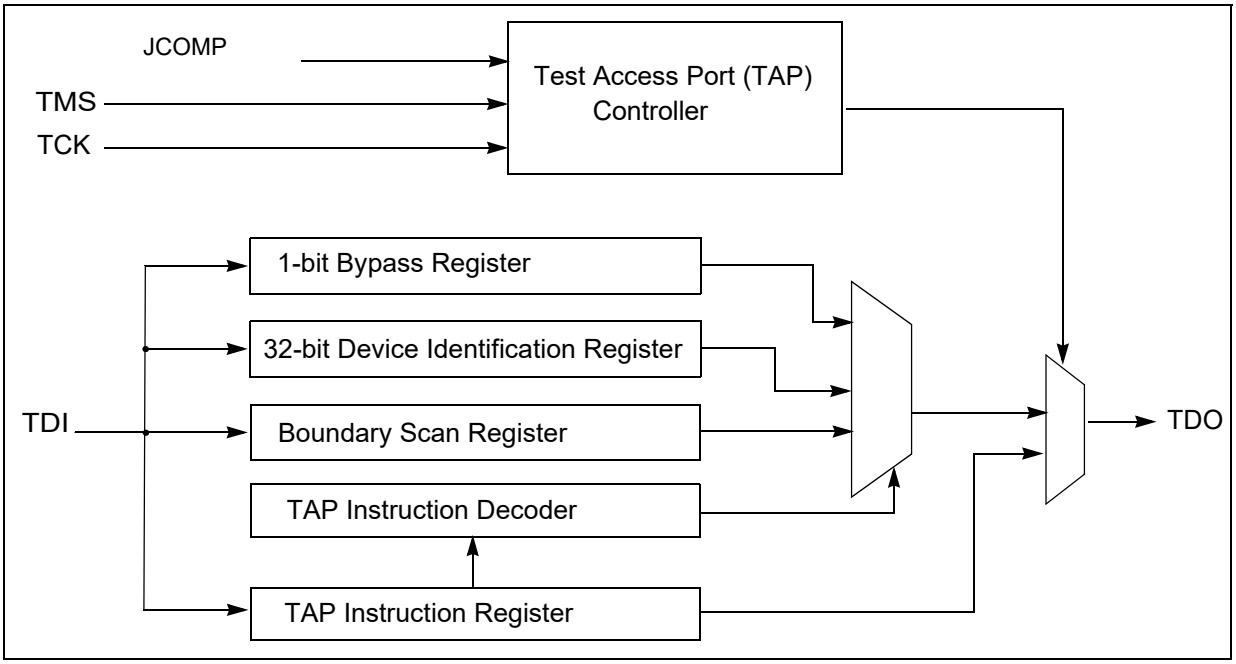
59.1 Introduction

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.

59.1.1 Block Diagram

The following is a block diagram of the JTAG Controller (JTAGC) block.

Figure 1035. JTAG (IEEE 1149.1) block diagram



59.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control and the ability to share the TAP.
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to *Table 987* 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.

59.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

59.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

59.1.3.2 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 59.4.4, JTAGC block instructions*.

59.1.3.3 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.

59.2 External signal description

The JTAGC consists of five signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 984. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z(1)	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

59.2.1 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.

59.2.2 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.

59.2.3 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 59.4.3, TAP controller state machine*.

59.2.4 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.

59.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

59.3 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.

59.3.1 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.

Figure 1036. 6-bit instruction register

	5	4	3	2	1	0
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

59.3.2 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.

59.3.3 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.

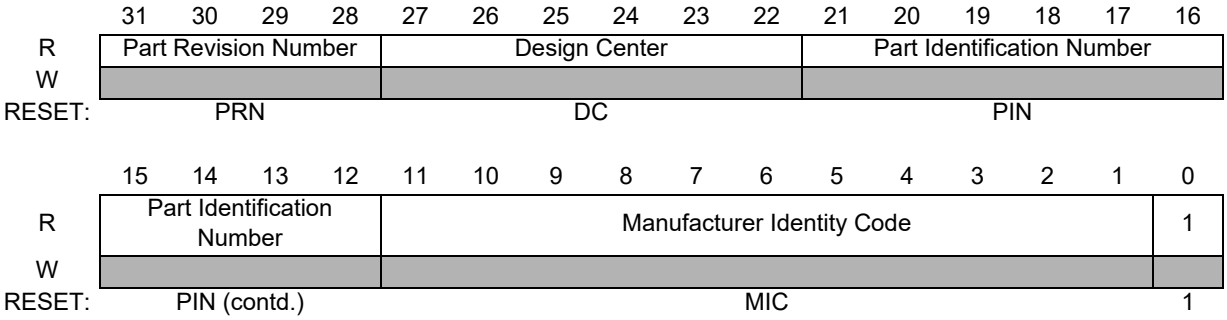


Figure 1037. Device identification register

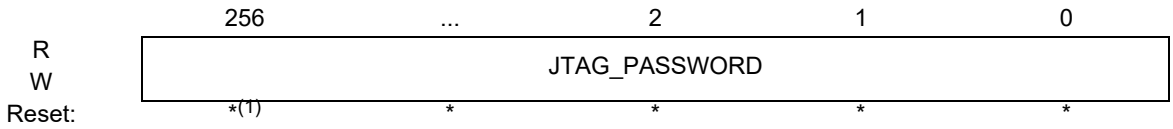
The following table describes the device identification register functions.

Table 985. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is 0x0.
DC	Design Center. Indicates the design center. Value is 0x2B.
PIN	Part Identification Number. Contains the part number of the device. Value is 0x306.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x020.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

59.3.4 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 1038. JTAG_PASSWORD register

The following table describes the JTAG_PASSWORD register functions.

Table 986. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

59.3.5 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 59.4.5, Boundary scan*. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

59.4 Functional description

This section explains the JTAGC functional description.

59.4.1 JTAGC reset configuration

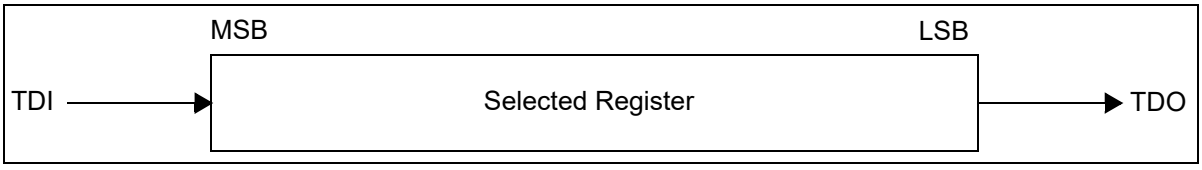
While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

59.4.2 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 59.4.4.8, ACCESS_AUX_x instructions*.

Data is shifted between TDI and TDO though 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 1039. Shifting data through a register

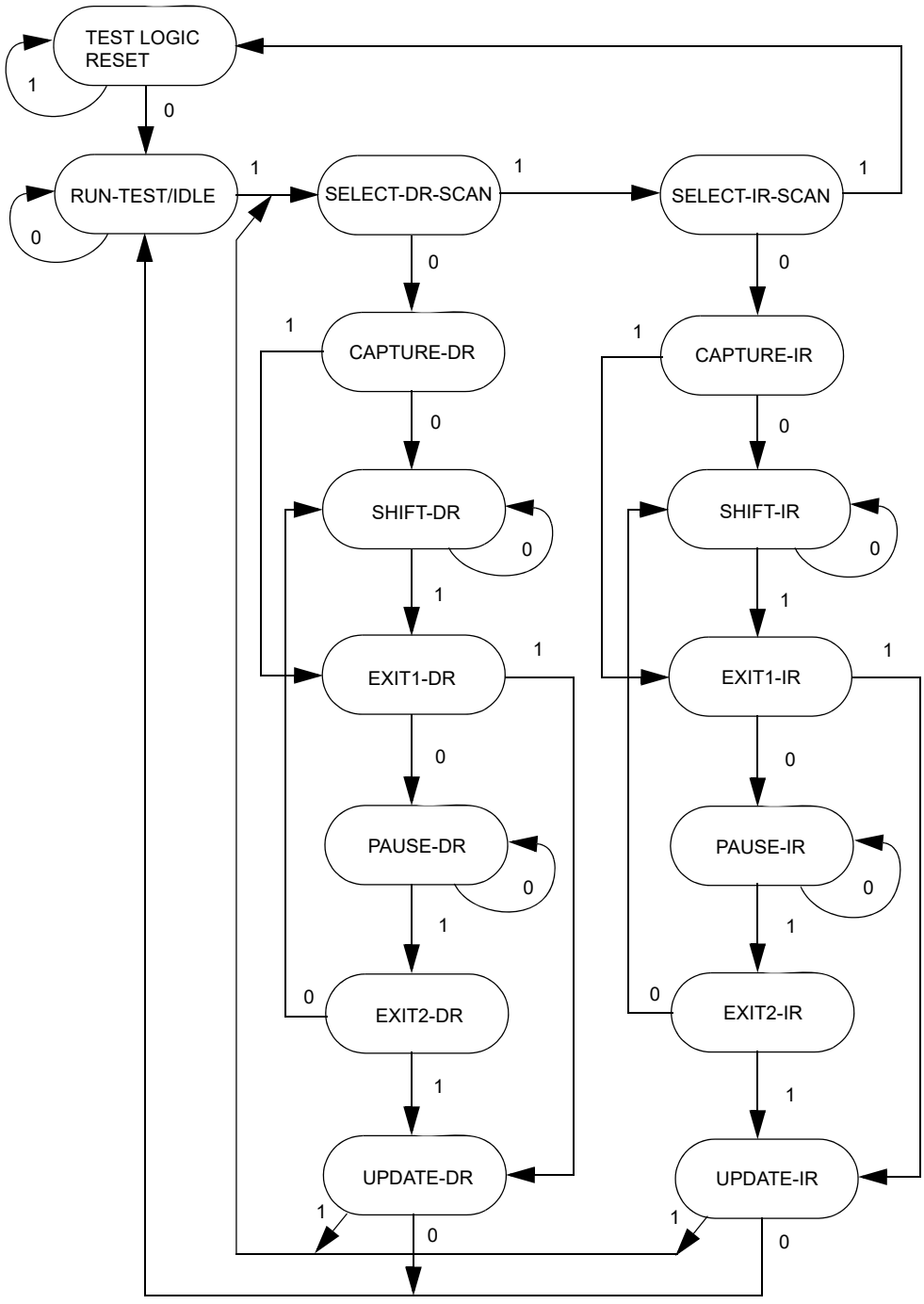


59.4.3 TAP controller state machine

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 1040. IEEE 1149.1-2001 TAP controller finite state machine



The value shown adjacent to each state transition represents the value of TMS at the time of a rising edge of TCK.

59.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

59.4.3.2 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.

59.4.4 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 987. 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
Factory debug reserved	000101	Intended for factory debug only
Factory debug reserved	000110	Intended for factory debug only
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register while three-stating all output pins and asserting functional reset
Factory debug reserved	001010	Intended for factory debug only
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_NR_PD	100001	Enables access to the NAR TAP controller on PD

Table 987. 6-bit JTAG instructions(Continued)

Instruction	Code[5:0]	Instruction Summary
Reserved	100010	Reserved
Reserved	100011	Reserved
Reserved	100100	Reserved
Factory test reserved	100101	Intended for factory test only
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_2	101010	Enables access to E200 Core 2
Reserved	101011	Reserved
Reserved	101100	Reserved
Reserved	101101	Reserved
Reserved	101110	Reserved
Reserved	101111	Reserved
Reserved	110000	Reserved
Reserved	110001	Reserved
Reserved	110010	Reserved
ACCESS_AUX_GTM	110011	Enables access to GTM TAP controller
ACCESS_AUX_BUS_MON_0	110100	Enables access to NXMC Client 0 TAP controller
ACCESS_AUX_BUS_MON_1	110101	Enables access to NXMC Client 1 TAP controller
Reserved	110110	Reserved
Reserved	110111	Reserved
Reserved	111000	Reserved
Reserved	111001	Reserved
ACCESS_AUX_SPU_A	111010	Enables access to SPU Client TAP controller
Reserved	111011	Reserved
ACCESS_AUX_PARALLEL(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

59.4.4.1 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.

59.4.4.2 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.

59.4.4.3 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.

59.4.4.4 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.

59.4.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

59.4.4.6 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.

59.4.4.7 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.

59.4.4.8 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.

59.4.4.9 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.

59.4.5 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.

59.5 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. Set the JCOMP signal to the JTAGC enable value, thereby enabling the JTAGC TAP controller
2. Load the appropriate instruction for the test or action to be performed

60 IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

60.1 References

- IEEE Std 1149.1-2001 (R2008), Standard Test Access Port and Boundary-Scan Architecture
- IEEE Std 1149.7, Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture

60.2 Introduction

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities. IEEE 1149.7 defines enhanced test and debug capabilities which are backwards compatible with IEEE P1149.1. This standard is also known as “Compact JTAG” or “CJTAG”.

The module implements the following parts of the IEEE 1149.7 standard:

- IEEE 1149.1 compliant behavior while allowing multiple on-chip TAP controllers
- Ability to assert test reset
- Improved system scan performance with one-bit IR and DR chip bypass paths
- Physical connection of a Debug and Test System (external tool) to a running system without corrupting its operation
- Operation with both 4-pin and 2-pin scan topologies

60.2.1 Types of operation

The 1149.7 standard provides a scalable and flexible solution to meet the needs of varied component and system complexities. There are three types of operation:

- Compliant
  - Compliant with the IEEE 1149.1 standard
  - Four-pin operation using the “Standard Protocol” — IEEE 1149.1 signaling where TCK, TMS, TDI and TDO signals transfer control/data information
  - More than one TAPC may be associated with the TAP (once made visible by private instructions)
- Extended
  - Compatible with the IEEE 1149.1 standard
  - Four-pin operation using the “Standard Protocol” — IEEE 1149.1 signaling where TCK, TMS, TDI(C) and TDO(C) signals transfer control/data information
  - Additional test/debug functions
- Advanced
  - Two-pin operation using the “Advanced Protocol” — signaling where TCK(C) and TMS(C) signals transfer control/data information
  - The TMS(C) signal is operated in a bidirectional manner with the TMS, TDI and TDO TAP information serialized and transferred via this signal

60.2.2 Deployment by class

The TAP.7 capabilities are deployed using five capability classes (T0–T4), each matching the needs of certain scan topologies and mixes of 1149.1 TAPs and 1149.7 TAPs. The classes provide the following capability:

- **Compliant Operation** – provides behavior compliant with the IEEE 1149.1-2001 standard:
  - **Class T0** – 1149.1 compliance when multiple TAPs are used on the same chip.
- **Extended Operation** – extends the capabilities of the IEEE 1149.1-2001 standard:
  - **Class T1** – T0 + Test access port power management, functional reset, and test reset.
  - **Class T2** – T1 + Chip bypass (1-bit) for IR/DR Scans + hot-connection with no errors.
  - **Class T3** – T2 + Star operation (TAPs are paralleled) with series equivalent scans for test.
- **Advanced Operation** – adds capabilities addressing test and debug needs in complex environments:
  - **Class T4** – T3 + Scan with two pins with the Advanced Protocol. A number of protocol optimizations maximize scan performance. TDI(C)/TDO(C) pins are optional and may have programmable function when implemented.

60.2.3 1149.7 TAP signals

The TAP signals for the T0–T4 TAP.7 classes are listed in *Table 988*. Signal names that end in “C” are associated with expanded functionality (above that provided by the IEEE 1149.1 standard).

The notation used in the table is as follows:

- M = Mandatory signal
- MC = Mandatory signal with expanded functionality
- O = Optional signal
- OC = Optional signal with expanded functionality

The CJTAG module provides T4 capability and implements the TCKC, TMSC, TDIC, TDOC, and nTRST pins.

Table 988. TAP.7 signal list

Signal Name	Description	Class				
		T4	T3	T2	T1	T0
TCK/TCKC	Test clock	MC	M	M	M	M
TMS/TMSC	Test mode select	MC	M	M	M	M
TDI/TDIC	Test data input	OC	MC	M	M	M
TDO/TDOC	Test data output	OC	MC	M	M	M
nTRST	Test reset	O	O	O	O	O
Mandatory signal count (minimum)		2	4	4	4	4

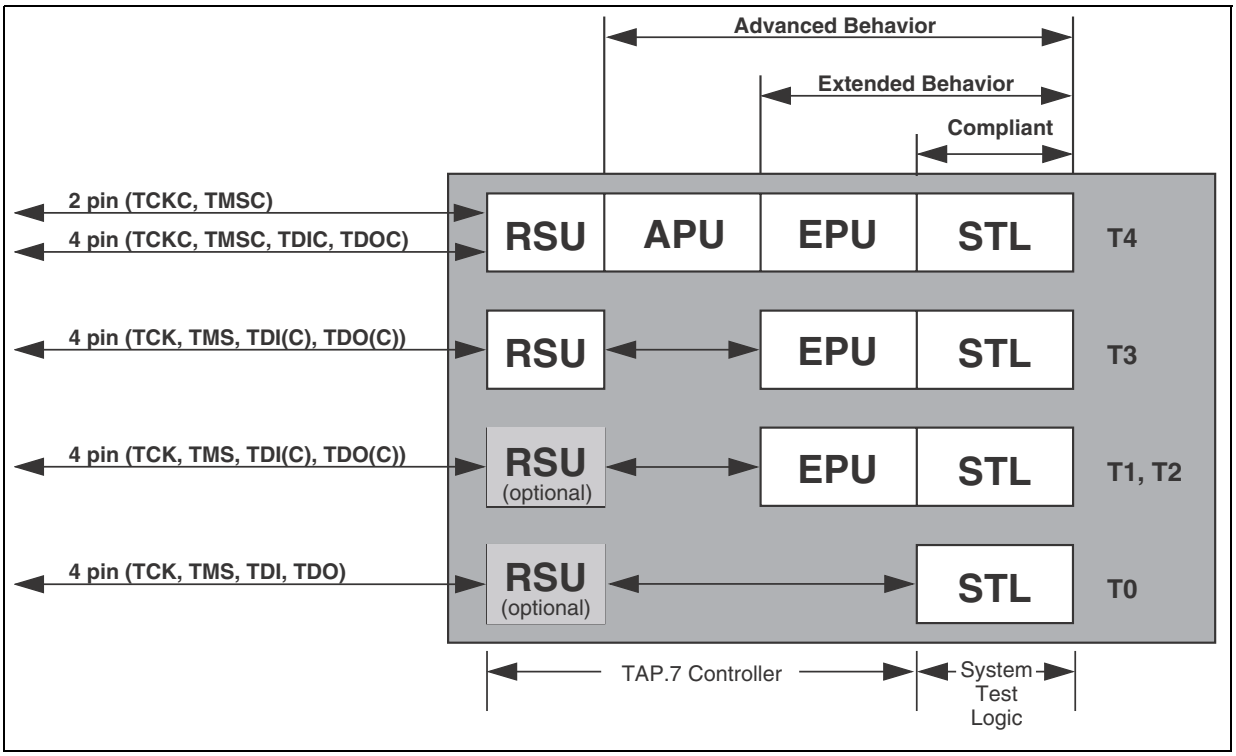
60.2.4 TAP.7 architecture

The TAP.7 hardware architecture is shown in *Figure 1041*. It is described with the hardware layers listed below:

- **STL – System Test Logic** – Logic with 1149.1 compliant behavior and underlying TAP hierarchy (T0 capabilities). Provides an IEEE 1149.1 interface for the T0 TAP.7. This logic is outside the CJTAG module.
- **RSU – Reset and Selection Unit** – A hardware layer that is placed between the APU, EPU, or STL and the TAP.7 signals (Added as an option to support the use of the Control Protocol) Provides reset and TAP.7 Controller selection services.
- **EPU – Extended Protocol Unit** – A hardware layer that is placed between the STL and the TAP.7 signals. (Added for T1, T2, and T3 capabilities). Provides an IEEE 1149.1 interface for the T1–T3 TAP.7s.
- **APU – Advanced Protocol Unit** – A hardware layer that is placed between the STL/EPU and the TAP.7 signals (added for T4 capabilities). Provides a T4 TAP.7 interface that is either narrow or wide, with the wide version providing an IEEE 1149.1 interface.

The CJTAG module supports T4 functionality that includes STL, RSU, EPU, and APU hardware layers.

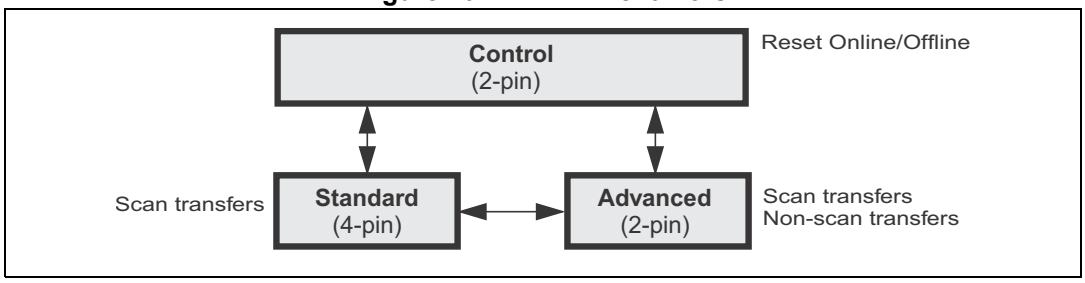
Figure 1041. TAP.7 Controller Architecture



60.2.5 Protocols

The TAP.7 architecture utilizes the three protocols shown in *Figure 1042*.

Figure 1042. TAP.7 Behaviors



The Standard Protocol transfers data with the TMS(C) signal value being sampled with a TCK(C) edge. The Advanced and Control Protocols use both data transferred with the TMS(C) signal value being sampled with a TCK(C) signal edge and escape sequences. The Standard Protocol may be used to switch to the use of the Advanced Protocol and vice versa. The Control Protocol can also be used to switch between the use of the Standard and the Advanced Protocol. The mandatory and optional deployment of these protocols with the TAP.7 classes is shown in *Table 989*.

Table 989. Class/protocol relationship

Class	Protocol		
	Standard	Control	Advanced
T0 – T2	Mandatory	Optional	N/A
T3	Mandatory	Mandatory	N/A
T4	Mandatory	Mandatory	Mandatory

The CJTAG module implements T4 functionality and supports the Standard, Advanced, and Control Protocols.

60.2.5.1 Standard Protocol

The Standard Protocol is the signaling defined by the IEEE 1149.1 standard. This protocol is also used to implement TAP.7 commands using only the TCK(C) and TMS(C) signals in a manner that does not require either TAP.7 controller instruction or data registers. The TAP.7 controller does not add bits to the scan paths of underlying technology. The Standard Protocol can be used to manage the TAP.7 controller functionality independent of the function associated with the availability of the TDI and TDO signals. The Standard Protocol supports STL data transfers only when these signals are present and provide the 1149.1 functionality defined by the IEEE 1149.1 standard.

60.2.5.2 Advanced Protocol

The Advanced Protocol supports both STL data transfers and management of the TAP.7 controller functionality with both the 2-signal (narrow) and 4-signal (wide) configurations. The Advanced Protocol serializes the TMS, TDI, and TDO information used with the Standard Protocol using only the TCK(C) and TMS(C) signals. A number of additional scan formats provide bit sequences optimized for specific use cases. Each of these additional scan formats affects the TMSC signaling sequences.

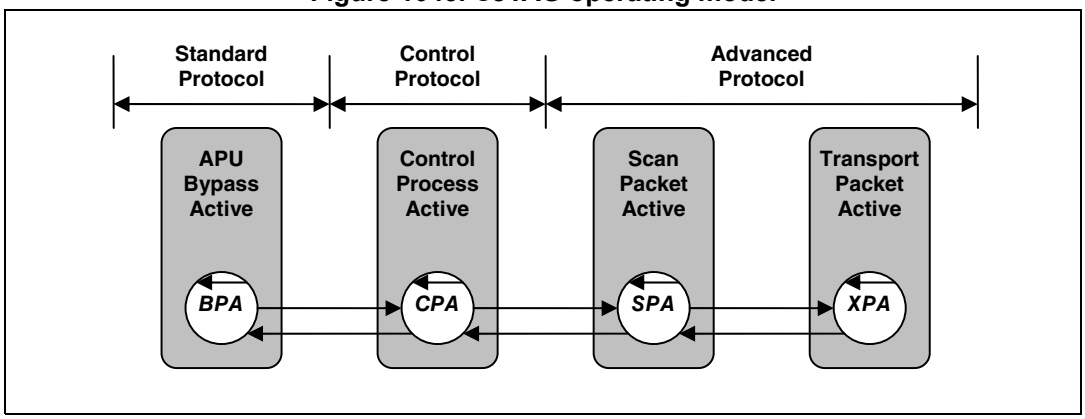
60.2.5.3 Control Protocol

The Control Protocol uses Escape sequences to reset the TAP.7 controller, place a TAP.7 controller offline or online, and indicate End of Transmission (EOT) with the Advanced Protocol. The characteristics of this protocol provide Hot-Connect-Protection by requiring a preamble of alternating ones and zeroes preceding a selection escape when a TAP controller is started up offline.

60.2.6 Operating models

Figure 1043 illustrates a simplified view of the CJTAG operating model.

Figure 1043. CJTAG operating model



60.3 CJTAG implementation summary

The CJTAG module is a T4 TAP.7 controller that implements the following functions.

60.3.1 T0 Functions

- IDCODE. - implemented in chip level TAPC (JTAGC module)
- 1149.1 Compliance at start-up
- Isolating / Excluding of embedded TAPs – implemented in chip level TAPC (JTAGC module)
- Operation with other technologies and scan topologies
- Multiple embedded TAPs

60.3.2 T1 functions

- EPU Command processing
- EPU Class-Specific Registers
- Asserting a test reset to the STL (TRESET Register)

60.3.3 T2 functions

- “Super Bypass” function – JScan0, JScan1, JScan2 scan formats. Allows the STL to be coupled or decoupled.

60.3.4 T3 functions

- Generation of a TAP.7 controller reset with a Reset Escape Sequence
- JScan3 Scan Format

60.3.5 T4 functions

- TMSC sampling on rising edge or falling edge of TCKC
- MScan (For test and debug)
- OScan 0-7
- TDIC and TDOC / Alternate pin function multiplexing (wide 4 only)

60.4 Ancillary services

60.4.1 Overview

Ancillary services are pertinent to any TAP.7 controller implemented with an RSU, such as the CJTAG module.

Ancillary services include:

- Resets
- Start-up options
- Escape sequences
- TAPC state machine

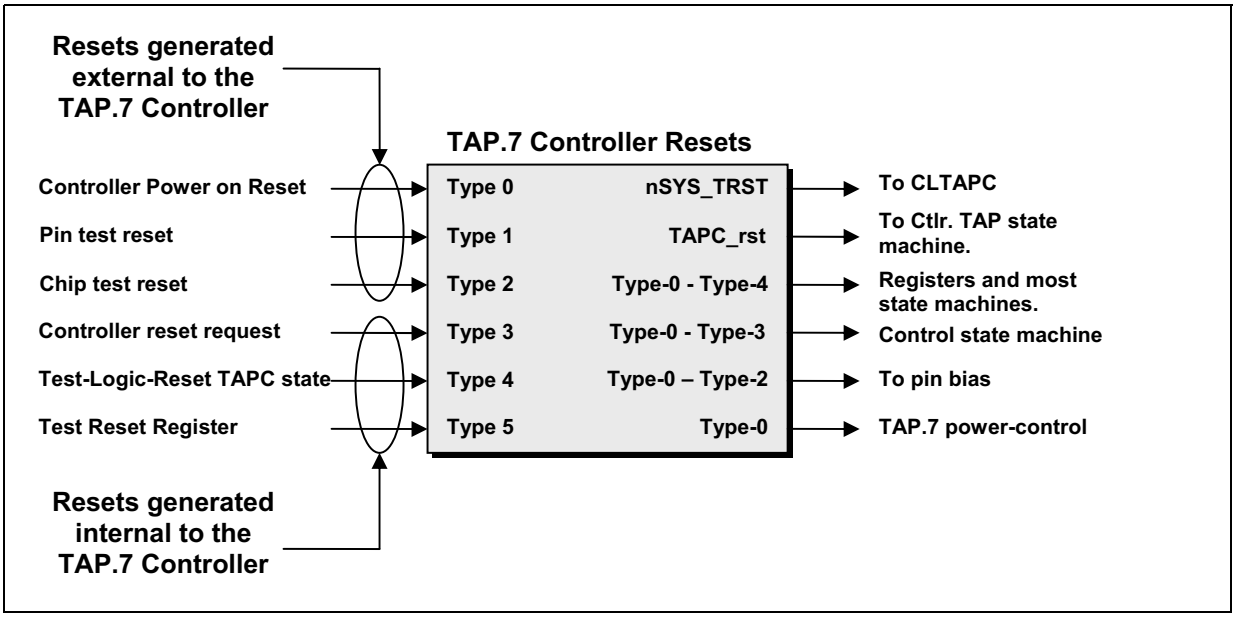
60.4.2 Resets

The TAP.7 controller reset function expands the reset function provided by the IEEE 1149.1 standard to provide additional reset types. It supports six reset types, Type-0 – Type-5, listed below. It can be implemented with as few as two reset types (Type-2 and Type-4), or as many as six reset types (all types) of reset inputs: This is determined by the class and options implemented.

The CJTAG module utilizes Type-0, Type-2, Type-3, Type-4, and Type-5 resets.

- **Type-0** – Reset of the TAP.7 by power management logic (power-on reset)
- **Type-1** – Chip level generated Test Reset. (equivalent to Type-2)
- **Type-2** – Externally generated Test Reset (nTRST/nTRST_PD pin)
- **Type-3** – TAP.7 controller generated reset
- **Type-4** – *Test-Logic-Reset* state
- **Type-5** – TAP.7 Controller TReset Register reset of the CLTAPC

Figure 1044. Conceptual view of the EPU reset logic and state machines



60.4.2.1 Type 5 reset

A Type-5 reset affects only the STL via the nSYS_TRST pin. It is asserted when the TRESET Register value is a logic 1. This register is set to its inactive state by reset Types 0-4.

60.4.2.2 Type 4 reset

A Type-4 reset is created by the *Test-Logic-Reset* state. It does not affect the TAP.7 TAPC state machine state as this state is the source of this reset, nor does it cause the assertion of the nSYS_TRST signal. It does however affect the coupling and decoupling of the STL. The Type-4 reset initializes the TAP.7 registers and most controller state machines. It affects but does not initialize the control state machine managing TAP.7 controller selection. It also establishes the default operating conditions.

60.4.2.3 Type 3 reset

A Type-3 reset is created by reset requests generated by the TAP.7 controller. A Type-3 reset performs the functions associated with a Type-4 reset and creates the *Test-Logic-Reset* state.

A Type-3 reset is generated in response to reset requests generated by EPU and APU logic

- A reset escape sequence
- Reset protocol sequences (delay packet directive, check packet directive)

A Type-3 reset request is generated with a reset escape sequence. This function is similar to but not the same as the reset function generated by a test reset pin. The Advanced Protocol includes certain bit patterns that initiate Type-3 resets. These bit patterns assist in the initialization of the TAP.7 controller when the TCKC signal is sourced by the TS and the external tool/TS connection is broken. The external tool may also generate these sequences, if desired, while the external tool/TS connection remains intact. These reset requests cause a Type-3 reset that is one clock wide.

With this being the case, consideration may be given to integrating the CJTAG module without the use of the Test Reset signal (nTRST/nTRST_PD pin).

60.4.2.4 Type 2 reset

A Type-2 reset is created by chip level logic to initialize the TAP.7 controller when TAP.7 power control is not implemented. It is unimplemented otherwise. It performs all of the functions of the Type-3 reset in addition to initializing the escape sequence detection logic.

60.4.2.5 Type 1 reset

A Type-1 reset is created with either the nTRST or nTRST_PD pin. It is not created when neither of these pins is implemented. It performs all of the functions of the Type-2 reset. It initiates TAP.7 controller power down provided the TAP.7 power control mode permits power down.

60.4.2.6 Type 0 reset

A Type-0 reset is implemented when TAP.7 controller power down is implemented. It performs all of the functions of a Type-1 reset and also initializes the TAP.7 power control logic.

60.4.2.7 Reset State Machine

The Reset State Machine assists in the translation of the reset inputs to the reset outputs. It handles Type-3 reset controller reset requests. The state machine state assures Type-0, Type-1, and Type-2 resets are forwarded to the STL without modifying their duration.

This state machine allows a Type-3 reset only when its state is “T3 Reset” (two TCK(C) falling edges have occurred in the absence of a Type-0, Type-1, and Type-2 reset).

Figure 1045. Reset state machine

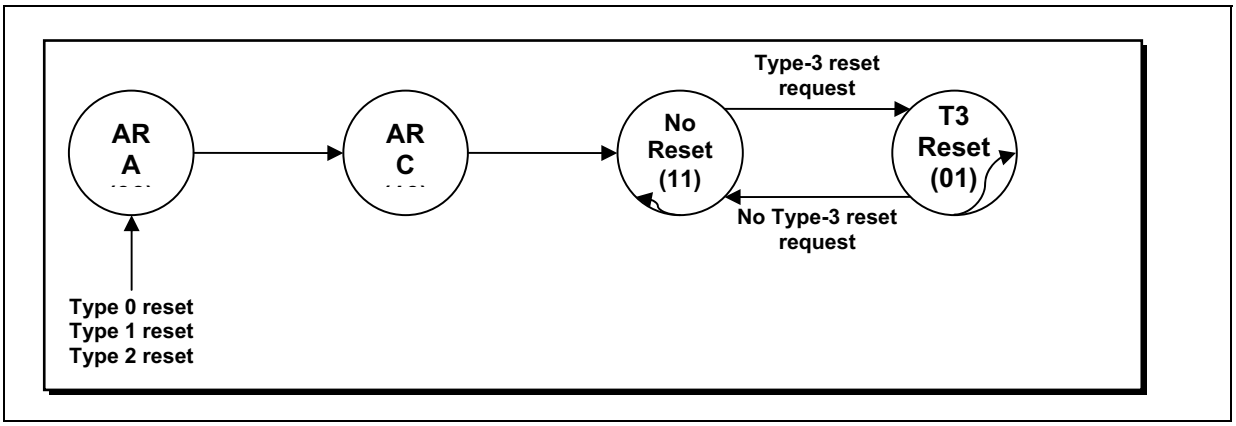


Table 990. Reset state descriptions

Reset state		Description
ARA	Async. Reset Asserted	An async. reset has occurred or is occurring
ARC	Async. Reset Completed	An async. reset has occurred and has been released
No Reset	No Reset	No Reset
T3 Reset	Type-3 Reset	A Type-3 reset request has been processed.

Type-0, Type-1, and Type-2 resets are fully asynchronous. These resets are passed directly to nSYS_TRST without modification. The Type-3, Type-4, and Type-5 resets are fully synchronous.

The effects of the TAP.7 controller reset types are shown in *Table 991*.

Table 991. Reset effects

Reset	CLTAPC	Pin-Hi-Z	State Machines other than CSM	Control State Machine (CSM)	TAPC SM State	Escape Sequence Detection	Power Control State
Type-5	Yes	—	—	—	—	—	—
Type-4	—	Yes	Yes	—	—	—	—
Type-3	Yes	Yes	Yes	Yes	Yes	—	—
Type-2	Yes	Yes	Yes	Yes	Yes	Yes	—
Type-1	Yes	Yes	Yes	Yes	Yes	Yes	—
Type-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes

60.4.3 Start-up Options

60.4.3.1 Overview

The TAP.7 start-up options are shown in *Table 992*. The only startup option currently supported by the CJTAG module is 1149.1 Compliant.

Table 992. Start-up Options

Start-up Option	Description
1149.1 Compliant	TAP.1 signal functionality, the CLTAPC is coupled
1149.1 Compatible	TAP.7 signal functionality, the CLTAPC is decoupled
1149.1 Protocol Compatible	No, or alternate TDIC/TDOC signal functionality, the CLTAPC is decoupled.
Offline-at-Start-up (Dormant)	The TAP.7 controller is offline and awaits synchronization to external tool operation so as to be placed online, the CLTAPC is decoupled after its state is moved to *Run-Test/Idle*

60.4.3.2 1149.1 Compliant start-up

Start-up behavior with the 1149.1 Compliant behavior start-up option:

- 1149.1-specific behavior / Standard protocol
- The TAPC finite state machine operated in lock step with the CLTAPC finite state machine.
- The TDIC and TDOC pins are present and provide the TDI and TDO function.
- The CLTAPC is coupled at start-up.

With this start-up option, the TAP.7 controller may be programmed and the scan topology may be interrogated. If the scan topology is known to be the Series Scan Topology at power-up, the TAP.7 can be operated as though it is a TAP.1.

Table 993. Reset values for TAP.7 start-up options

Start-up Option	Type-0 – Type-3 Reset?	Default TAP.7 and TAP.7 controller characteristics				
		TAP.7 Controller online	CLTAPC Coupled	SGC Reg. == 1	SCNFMT	Initial TMSC pin bias
1149.1 Compliant	x	Yes	Yes	Yes	JScan0	PU

60.4.4 RSU operation

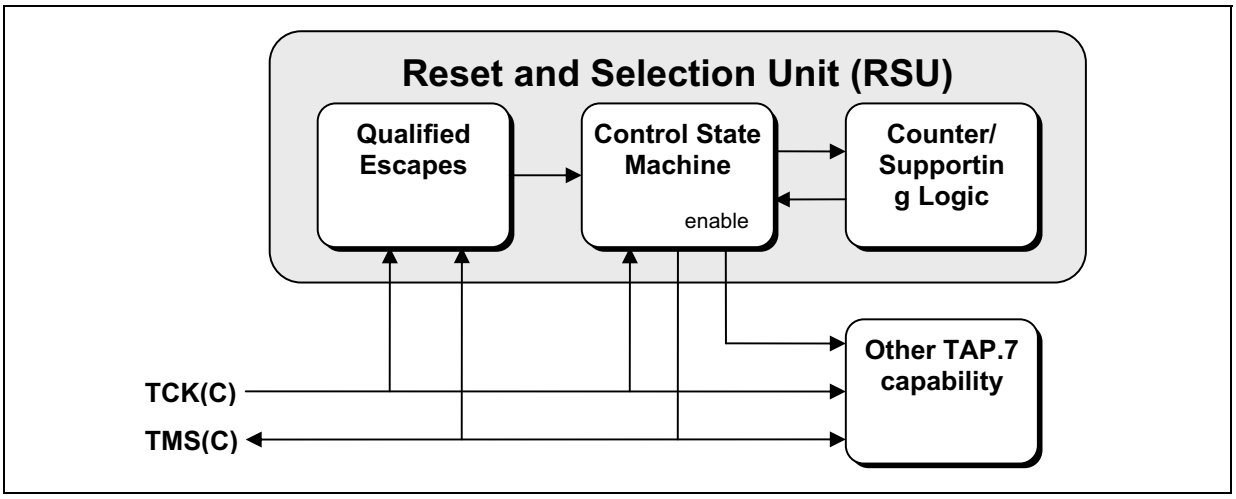
The RSU uses TCKC and TMSC with the control protocol for configuration control.

60.4.4.1 General Operation

From a very high level, the RSU enables the remainder of the TAP.7 functionality when the TAP.7 controller is Online and disables the remainder of the TAP.7 functionality otherwise. The RSU either consumes or discards the TMS(C) information while the TAP.7 controller is

Offline. The relationship of the escape sequence detection, technology selection logic, and the underlying technology is shown in *Figure 1046*.

Figure 1046. Conceptual view of technology selection mechanism



60.4.4.2 Common Signaling Across Technologies

With the TAP.7 architecture, the signaling conventions chosen for technology selection is multiple data transitions of the TMS(C) signal while the TCK(C) signal is a logic 1. The common signaling space is available when the external tool sources the TCK(C) signal as it is capable of creating these signaling conventions. It is unavailable when the TS sources the TCK(C) signal as the external tool cannot stop the TCKC signal at a logic 1 to create these signaling conventions.

60.4.4.3 Escape Sequence Detection

TAP.7 TAPC interprets the count of the number of TMS(C) edges while TCK(C) is a logic 1 as one of four escape sequences. Each escape sequence has a different function and TMSC edge count. These escape sequences, their edge counts, and their functions are described below:

- **Custom** (2 or 3 edges)– Ends scan and transport transfers with a TAP.7 controller, may be used for other purposes with another technology.
- **Reset** (> 7 edges)– Resets all technologies (generates a Type-3 TAP.7 TAPC reset)

The external tool creates an escape sequence by generating one or more TMS(C) edge pairs while the TCK(C) signal is a logic 1 value. Even and odd TMS(C) edge counts beginning with two are given the same meaning.

An escape sequence:

- Can be detected only when a Type-0, Type-1 or Type-2 reset is **not** asserted.
- Begins and ends while the TCK(C) is a logic 1.
- Uses TMS(C) as a clock while the TCK(C) is a logic 1.
- Overlays additional control information onto the normal information that is transferred with the TCK(C) and the TMS(C) without changing the normal information.
- Must be detected when TCK(C) is a logic 1 regardless of the TCK(C) and TMS(C) drive histories.

60.4.4.3.1 Custom escape sequence

An online technology can use the custom escape sequence in any manner it chooses. It is used as the End of Transmission (EOT) of certain data types (formats OScan4-7) with T4 and above TAP.7s.

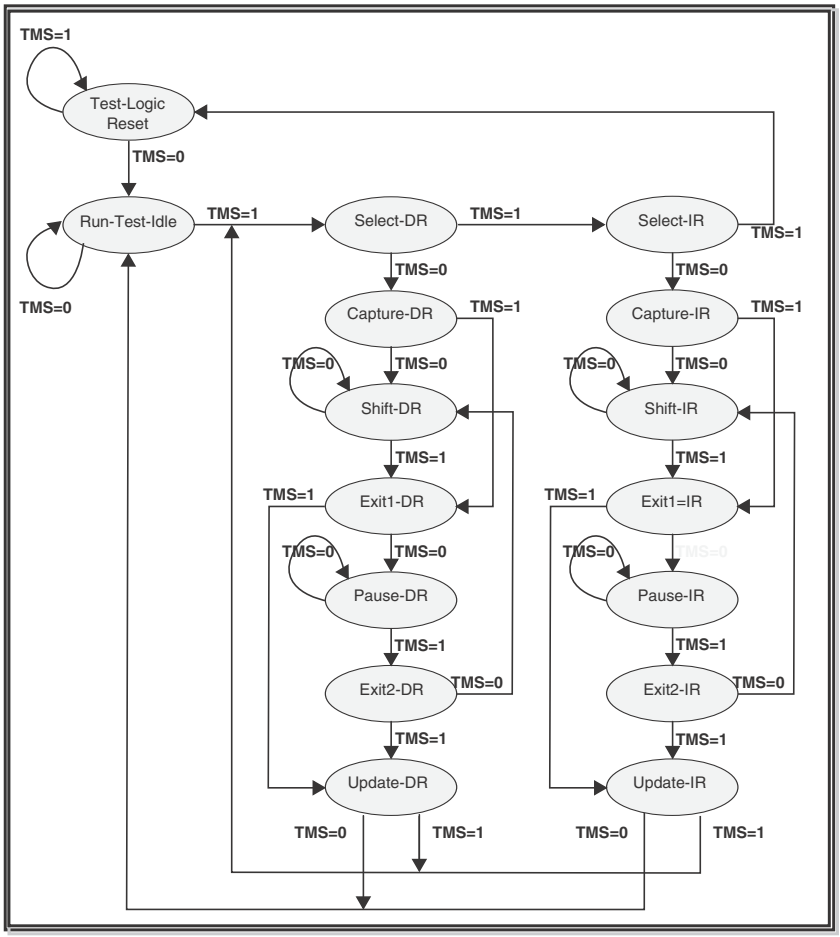
60.4.4.3.2 Reset escape sequences

A reset event initializes all technologies sharing the TCK(C) and TMS(C) connectivity. A reset escape sequence resets the technology. It is recommended that this reset occur with the falling edge of TCKC. With a TAP.7 controller a reset escape sequence generates a Type-3 reset beginning with the falling edge of TCK(C) following the asynchronous detection of the event.

60.4.5 TAPC State Machine

T1 and above TAP.7s require knowledge of the TAPC state machine state. With T2 and above TAPs this knowledge must be developed independently of the CLTAPC. This is generally accomplished by implementing a TAPC state machine within the EPU. This TAPC state machine provides the function of the state machine within a standard 1149.1 TAPC. It is recommended that this state machine is implemented to change state on the falling edge of TCK(C) as this has numerous advantages to a TAP.7 controller. The EPU does not contain 1149.1 Instruction or Data Registers. The APU controls the state changes of this TAPC state machine with T4 and above TAP.7s.

Figure 1047. Falling edge TAP controller state machine



60.5 EPU (Extended Protocol Unit) Operation

60.5.1 EPU Operation

The EPU adds functionality to the BYPASS and IDCODE instructions.

60.5.1.1 Zero-bit DR-Scans (ZBS)

A zero-bit DR-Scan (ZBS) is a TAPC-state sequence that begins with the *Select-DR-Scan* state and ends with an exit from the *Update-DR* state without an intervening *Shift-DR* state. The external tool controls the EPU using ZBSs.

The use of ZBSs can begin immediately following the *Test-Logic-Reset* state since this state initializes the Instruction Registers of TAPs with either the *BYPASS* or *IDCODE* instructions.

60.5.1.2 Utilizing ZBSs for TAP.7 Controller Functionality

Beginning with a ZBS count of zero, the ZBS count is incremented with each consecutive occurrence of a ZBS without encountering a *Shift-DR* state. This count cannot be

incremented past seven. *Run-Test/Idle* is the only state that may occur between consecutive occurrences of ZBSs without terminating the count of consecutive ZBSs.

60.5.1.3 Locking the ZBS Count

When a DR-Scan containing a *Shift-DR* state occurs, and the ZBS count is greater than zero, the ZBS count is not incremented when the *Update-DR* state is reached. It is instead locked at its current value. ZBSs occurring after locking the ZBS count do not affect the locked ZBS count. Locking the ZBS count is the equivalent of storing the count for subsequent use.

60.5.1.4 Control levels

Locking the ZBS count activates a control level that is equal to the locked ZBS count (1-7) when ZBSs are being used for TAP.7 functionality and merely locks the ZBS count otherwise.

1. ZBSs may be used by the STL or another function
2. TAP.7 command generation
3. Reserved control level
4. Accesses to optional TAP.7 scan paths
5. Force offline operation (optional)
6. External tool uses

Control level one may be used by the STL since there is no TAP.7 controller functionality associated with this control level.

Control level two enables DR-Scans to create TAP.7 commands. A portfolio of these commands supports the deployment of TAP.7 controllers in the Series or Star Scan Topologies. These commands manage TAP.7 controller registers and perform other functions.

Control level three is reserved for use with a subsequent specification revision.

Control levels four and five provide access to custom (private) scan paths when access to these paths is permitted by a TAP.7 register bit used for this purpose. The EPU provides a 1-bit scan path otherwise. No other EPU functionality is associated with these control levels. The CJTAG module does not support the use of this control level.

Control level six may be optionally used to force offline operation with a TAP.7 controller that supports this capability. The EPU provides a 1-bit scan path for this control level. The CJTAG module does not support the use of this control level.

Control level seven is reserved for external tool use. With this control level, external tool capabilities may be managed using the same infrastructure used to manage TAP.7 controller capability. The EPU provides a 1-bit scan path for this control level.

60.5.1.5 Exiting a Control Level

A control level is exited when one of the following occurs:

- The *Select-IR-Scan* TAP.7 controller state
- The *Test-Logic-Reset* state
- Certain TAP.7 controller commands (Exit Control Level – ECL register bit)
- An event that synchronizes the operation of TAP.7 controllers

60.5.1.6 Zeroing the ZBS Count

Events causing an exit from a control level also zero the ZBS count. Scan Selection Directives (SSDs) associated with T3 and above TAP.7s zero the ZBS count when the control level has not been locked.

60.5.1.7 Utilizing ZBSs within the STL

After the *Test-Logic-Reset* state, ZBSs are used for basic EPU control. ZBSs that are intended for TAP.7 use are henceforth called "***TAP.7 ZBSs***". ZBSs that are intended for other uses are henceforth called "***Other ZBSs***". Two TAP.7 registers enable the STL's use of ZBSs. Storing these registers causes a control level exit, zeroes the ZBS count, and enables the use of Other ZBSs. Storing the Suspend Register (**SUSPEND**) suspends the use of control levels. Storing the ZBS Inhibit Register (**ZBSINH**) inhibits ZBS detection.

When either of these methods is used, the *Test-Logic-Reset* state enables the use of TAP.7 ZBSs in addition to initializing the TAP.7 controller.

The operation of the TAP.7 controller changes with the use of these two paths.

When the System Path is used:

- A count of eight consecutive ZBSs is used to create a logic 0 SUSPEND Register value.
- The CLTAPC supplies the TAP.7 scan path and controls the drive of TDO(C), provided it is not bypasses.
- The EPU supplies the TAP.7 scan path (a one bit bypass) when the CLTAPC is bypassed.
- Functions supporting Boundary Scan in a Star Scan Topology are enabled.

When the EPU Path is used:

- When the control level has not been locked, the TDO(C) signal remains high-impedance.
- The command control level may be created
- When commands are used, the TDO(C) signal remains high-impedance except during *Shift-DR* states within CR-Scans, and remains high impedance at other times.
- Auxiliary scan paths may be accessed with control levels four and five.

60.5.2 EPU Registers

60.5.2.1 Global

A global register is a register whose value is stored at the same time and with the same value as registers with the same name in all TAP.7 TAPCs sharing an external tool connection. They are stored using TAP.7 TAPC commands dedicated for this purpose. These commands store a single global register. Global registers manage TAP.7 TAPC functions that affect the synchronized operation of a TAP.7 TAPC. An example of a global register is the Scan Format (SCN_FMT) Register used to define the TAPC signal protocol.

A second means of storing all global registers is available. The selection of the TAP.7 TAPC with the technology selection function stores either the Scan Format register value (short form) or all global registers (long form). Storing all global registers provides a simple means to synchronize the values of all global registers when Technology Selection causes a non-operating TAP.7 TAPC to become operational while other TAPCs are operational.

60.5.2.2 Local

Local registers manage TAP.7 TAPC functions that do not affect the synchronized operation of TAP.7 TAPCs. Local registers are conditionally stored using TAP.7 controller commands dedicated for this purpose. These commands provide for changing the value of these registers one a time with a unique value in a Series Scan Topology based on the TAPC's position on the scan chain. Local registers are stored with a unique value in a Star Scan Topology based on a TAPC address assigned to the TAPC.

Registers programmed with commands are described in this section. The CJTAG module implements all mandatory T4 registers as well as the optional APFC, TRESET, and RDBACK0 registers.There are 16 mandatory and 10 optional registers that can be grouped in classes as shown in Read-only – Configuration and register read back.

- **Control** – TAP.7 controller behaviors and characteristics.
- **Options** – Control of optional functions.
- **Select** – Controls the selection of a TAP.7 for Scan and execution of conditional commands.
- **Read-only** – Configuration and register read back.

Table 994. TAP.7 controller register list (managed with commands)

Register type	Global/ local	Width	Register mnemonic	Name
Control	Global	1	ECL	Exit Control Level
		1	SUSPEND	Suspend
		1	ZBSINH	ZBS Detect Inhibit
		5	SCNFMT	Scan Format
		1	SSDE	SSD Enable
		2	DLYC	Delay control
		2	RDYC	Ready Control
Options	Local	1	TRESET	Test Reset
		2	APFC	Aux. Pin Func. Cntl.
Select	Local	1	CGM	Cond. Group Member
		1	SGC	Scan Group Candidate
		1	SEDGE	Sampling Edge
Read-only	Local	32	RDBACK0	Read-back 0
		32	CNFG0	Configuration Register 0
M = Mandatory register O = Optional register				

60.5.2.3 Registers

60.5.2.3.1 Exit Control Level (ECL)

Storing this register zeroes the ZBS count and unlocks the control level.

This register is set using the STMC command. It remains logic 1 only momentarily before immediately returning to logic 0.

This register is cleared by a global register load.

60.5.2.3.2 Suspend (SUSPEND)

Setting this register suspends the use of control levels. When this register is a logic 1, the “System Path” (STL) is used.

- **1** – Suspend use of control levels. Enables “System Path.”
- **0** – Enable use of control levels.

SUSPEND is set with the STMC command.

This register is cleared when 8 consecutive ZBSs are detected without the locking of the ZBS count.

This register is cleared by a global register load.

60.5.2.3.3 ZBS Inhibit (ZBSINH)

Storing to ZBSINH inhibits ZBS detection.

- **1** – Inhibit ZBS detection.
- **0** – Enable ZBS detection.

ZBSINH is set with the STMC command.

This register is cleared by a global register load.

60.5.2.3.4 Scan Format (SCNFMT[4:0])

Set using Store Scan Format (STFMT) command.

Read using Scan String (SCNS) command.

Table 995. Scan Format Register Values

TAPC.7 Class	SCNFMT	Description
T2–T4	00000	JScan0: – 1149.1 Compliant Operation. – Requests the coupling of the CLTAPC when the Run-Test/Idle state is reached.
	00001	JScan1: – 1149.1 Compatible Operation – Requests the decoupling of the CLTAPC when the Run-Test/Idle state is reached.
	00010	JScan2: – 1149.1 Compatible Operation. – Requests the coupling or decoupling of the CLTAPC when the Run-Test/Idle state is reached based on the value of the CACT register
T3–T4	00011	JScan3: – Non-compatible 1149.1 Operation – Allows the detection of a T2 TAP.7 controller that is deployed in a Star-4 topology. – Enables TDOC signal behavior that prevents drive conflicts in a Star-4 Scan Topology. – Enables the use of SSDs
T4	01000	OScan0 – Data rate dependent components – Single TAP.7 communication – Debug and Test applications
	01001	OScan1 – TAP.1 components – Single/Multi-TAP.7 communication – Debug and Test applications
	01010	OScan2 – TAP.1 components – No TDI or TDO in non-shift states – Debug applications
	01011	OScan3 – TAP.1 components – No TDI in non-shift states – No TDO in any state

Table 995. Scan Format Register Values(Continued)

TAPC.7 Class	SCNFMT	Description
T4	01100	OScan4 – Data rate dependent components – Single TAP.7 communication – Debug and Test applications
	01101	OScan5 – TAP.1 components – Single/Multi-TAP.7 communication – Debug and Test applications
	01110	OScan6 – TAP.1 components – No TDI or TDO in non-shift states – Debug applications
	01111	OScan7 – TAP.1 components – No TDI in non-shift states – No TDO in any state
	10000	MScan – Data rate dependent components – Multi TAP.7 communication – Directed CID assignment – Virtually any IP with compliant or non-compliant 1149.1 behavior
	10001 – 11111	Reserved

The use of a Scan Format value other than 00000b – 10000 shall not change the value of the SCNFMT Register.

60.5.2.3.5 Scan Selection Directive Enable (SSDE)

This register specifies whether the Scan Selection State may be managed using the scan selection directives.

- **0** – Scan selection directives are not enabled.
- **1** – Scan selection directives are enabled.

This register is set using the STMC command.

This register is read using SCNS (Scan String), RDBACK0.

60.5.2.3.6 Delay Control (DLYC[1:0])

The Delay Control register provides a means for the external tool to delay the completion of an SP. This delay is essentially an external tool stall. This delay may be zero, one, two, or n TCKC periods. This delay is especially useful when the TCKC is sourced by the TS and a Standard-to-Advanced Protocol adapter is added to an external tool supporting only the Standard Protocol. It allows an external tool design additional time to receive TDO, advance the external tool TAPC state, receive TMS and TDI information, and begin generation of a

new SP payload. The DLYC register should be programmed before the use of the Advanced Protocol begins after the use of the Standard Protocol.

Set using the STMC command.

Read using SCNS, RDBACK0

Bits 18-17 of the “Global Register.”

60.5.2.3.7 Ready Control (RDYC[1:0])

The Ready Control register defines behavior exhibited in the SP payload output bit frames when the STL stall opportunities (RDY bits) are included as control information. This register defines the number of additional bits inserted in these bit frames. These bits provide more time for a high-performance external tool to ascertain whether the TAP.7 controller is ready to complete the SP payload. The use of these bits makes a high-performance external tool design easier as input and output buffer delays play less of a role in limiting the TAP.7 controller performance. The RDYC register should be programmed before the use of the Advanced Protocol begins.

Set using the STMC command.

Read using SCNS, RDBACK0.

Bits 18-17 of the Global register state.

60.5.2.3.8 Test Reset (TRESET)

Asserts a test reset to the STL.

- **0** – The nSYS_TRST and SYS_TMS signals presented to the STL is not influenced by this bit.
- **1** – The nSYS_TRST presented to the STL is asserted and the SYS_TMS signal is a logic 1.

Set using the STC1 command.

The effects of the TRESET Register may be modified by a private TAP.7 controller register. When this private register is unimplemented or has a value of zero, the TRESET register causes the maximum initialization of the STL with nSYS_TRST and the SYS_TMS logic 1 value. Otherwise, the value of the private register may modify the function of the TRESET Register bit.

When the TRESET Register is a logic 1, the STL scan path appears broken as the state of CLTAPC remains *Test-Logic-Reset* while the EPUTAPC state progression continues. When the operation of the EPUTAPC and CLTAPC controllers is coupled, as with a T1 TAP.7, the state of these two TAPCs may be resynchronized to the *Run-Test/Idle* state by:

- An STC1 Command that sets the TRESET Register value to a logic 0.
- A stay in the Run-Test/Idle state of two or more SYS_TCK periods immediately following the Update-DR state of this command

A failure to follow the guidelines is considered a programming error and will result in erroneous system operation. The EPUTAPC and CLTAPC states will not be synchronized as they progress through the state diagram.

60.5.2.3.9 Auxiliary Pin Function Control (APFC[1:0])

This register enables TDI and TDO alternate functions when 2-pin TAP.7 operation is selected.

- **0x** – TDI and TDO have JTAG function.
- **1x** – TDI and TDO have alternate function.

Set using STC2 command.

Read using SCNS, RDBACK0.

60.5.2.3.10 Conditional Group Member (CGM)

Used by STC1, STC2, STTESTM, and EXC3 – EXC0 commands as the “condition” for updating registers.

Set and cleared using MCM and SCNB commands.

Read using SCNS, RDBACK0.

60.5.2.3.11 Scan Group Candidate (SGC)

This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used.

- **0** – Decoupling action.
- **1** – Coupling action.

Set and cleared using STMC, MSC, and SCNB commands.

Read using SCNS, RDBACK0.

60.5.2.3.12 Sampling Edge (SEDGE)

The Sampling Edge register defines the TCKC edge used to sample the TMSC input.

- **0** – Sample TMSC on the TCKC falling edge.
- **1** – Sample TMSC on the TCKC rising edge.

Set and cleared using STC1 command.

Read using SCNS, RDBACK0

60.5.2.3.13 Read Back 0 (RDBACK0)

Non-transport related register information.

Read using SCNS command.

Unimplemented registers shall read back as “0”.

Table 996. RDBACK register format

Bit	Width	Register Mnemonic	Name
4:0	5	SCNFMT	Scan Format
6:5	2	PWRMODE	Power-control Modes
7	1	FRESET	Functional Reset
8	1	TRESET	Test Reset

Table 996. RDBACK register format(Continued)

Bit	Width	Register Mnemonic	Name
9	1	SGC	Scan Group Candidate
		CGM	Conditional Group Member
11	1	SSDE	Scan Selection Directive Enable
13:12	2	TOPOL	Topology
14	1	SREDGE	Sampling Edge
16:15	2	DLYC	Delay Control
18:17	2	RDYC	Ready Control
20:19	2	APFC	Auxiliary Pin Functional Control
22:21	2	STCKDC	SYS_TCK Duty Cycle
31:23	11	Reserved	Read as a logic 0

60.5.2.3.14 Configuration Register 0 (CNFG0)

Table 997. Configuration Register 0 Format

Field	Bit #	Width	Name	Description
CNFG (Class)	3:0	4	CLASS[3:0]	TAP.7 Class
				0000 – TAP.1 or TAP.7 Class T0
				0001 – Class T1
				0010 – Class T2
				0011 – Class T3
				0100 – Class T4
				0101 – Class T5
CNFG (Revision)	7:4	4	REV[3:0]	TAP.7 specification revision. Initial revision = all zeroes (0s)
				0000 – Reserved
				0001 – IEEE Std 1149.7-2008
				0010 – 1111 - Reserved
CNFG (Format)	11:8	4	CNFGFMT[3:0]	Configuration Register Format
				xxx1 – CNFG0[31:12] are implemented
				xx1x – CNFG1 implemented
				x1xx – CNFG2 implemented
				1xxx – CNFG3 implemented

Table 997. Configuration Register 0 Format(Continued)

Field	Bit #	Width	Name	Description
T1 Options	12	1	RDBKS	RDBACK supported
				0 – Unsupported
				1 – Supported
	13	1	TRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	14	1	FRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	17:15	3	PWRMODES[2:0]	PWRMODES[x] – Power Control Mode x suported where x = 0,1,2
				0 – Unsupported
				1 – Supported
	18	1	PCMR	PCMR – PM default POWRMODE value
				0 – No
				1 – Yes
T2 Options	19	1	DCAS	DCAS – Decouple at start-up
				0 – No
				1 - Yes

Table 997. Configuration Register 0 Format(Continued)

Field	Bit #	Width	Name	Description
T4 Options	21:20	2	SSCANS[1:0]	x0 – SScan1:0 Unsupported
				x1 – SScan1:0 Supported
				0x – SScan3:2 Unsupported
				1x – SScan3:2 Supported
	24:22	3	OSCANS[2:0]	xx0 – OScan3:2 Unsupported
				xx1 – OScan3:2 Supported
				x0x – OScan5:4 Unsupported
				x1x – OScan5:4 Supported
				0xx – OScan7:6 Unsupported
				1xx – OScan7:6 Supported
	25	1	APFCS	Auxiliary Pin Function Control supported
				0 – Unsupported
				1 – Supported
	26	1	TAPWIDS	TAP Width Default
				0 – 2-pin supported
				1 – 4-pin supported
	27	1	TAPWLCK	TAP Width Locked
				0 – Not locked
				1 - Locked
	28	1	STCKDCS	SYS_TCK Duty Cycle supported
				0 – Unsupported
				1 – Supported
Reserved	31:29	3	Reserved	Reserved, read as zero

60.5.2.3.15 Register reset values

Table 998. Reset values of TAP.7 controller registers managed with commands

Register type	Width	Register mnemonic	Name	Values for reset type			
				0	1 – 3	4	5
Control	1	ECL	Exit Control Level	0	0	0	NC
	1	SUSPEND	Suspend	0	0	0	
	1	ZBSINH	ZBS Detect Inhibit	0	0	0	
	5	SCNFMT	Scan Format	DF	DF	DF	
	1	SSDE	SSD Enable	0	0	0	
	2	DLYC	Delay control	00	00	00	
	2	RDYC	Ready Control	00	00	00	
Options	1	TRESET	Test Reset	0	0	0	NC
	2	APFC	Aux. Pin Functional Cntl.	DP	DP	NC	
Select	1	CGM	Conditional Group Member	0	0	0	NC
	1	SGC	Scan Group Candidate	DSGM	DSGM	DSGM	
	1	SREDGE	Sampling Rising Edge	0	0	0	
Read-only	32	RDBACK0	Read-back 0	N/A	N/A	N/A	N/A
	32	CNFG0	Configuration Register 0				

NC = No Change

DF = Default Scan Format

DM = Default Power Control Mode

DCA = Default Coupling Action

DT = Default Topology

DP = Default Pin Function

60.5.3 EPU Commands

There are two command types:

- Two-Part Commands
- Three-Part Commands

Any number of two-part and three-part commands may be issued in any combination before the command control level is exited. If the command control level is exited before a two- or three-part command is completed, the command is aborted and becomes a NOP.

60.5.3.1 Two-Part Commands

TAP.7 controller commands are 10-bit values. The 10-bit commands are created by two consecutive DR-Scans when the control level is locked at the command control level (two). The first DR-Scan provides a five-bit command opcode. The second DR-Scan provides a five-bit operand. These two scans are called the command part one (CP1) and command part two (CP2), respectively. A command created entirely with these two scans is called a two-part command.

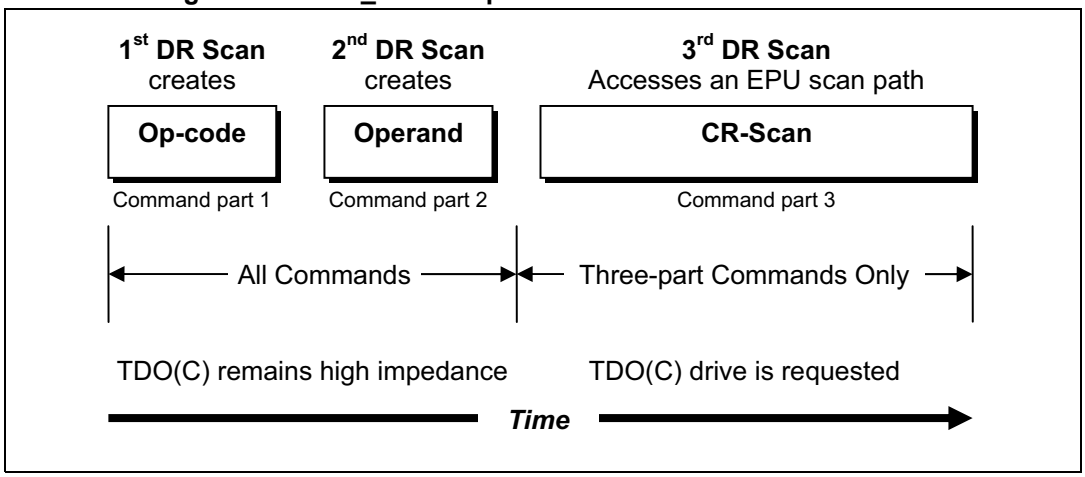
60.5.3.2 Three-Part Commands

Some commands are used to send and/or receive data values other than values embedded in the command's operand. These commands are called three-part commands. With these commands, CP1 and CP2 are followed by an additional DR-Scan to transport a data value to or from an EPU scan path. When used for this purpose, the DR-Scan is called a Control Register Scan (CR-Scan). The CR-Scan is a minimum of zero bits in length with there being no maximum length. The CR-Scan path is described in SECTION.

60.5.3.3 Command Sequence

The two 5-bit values concatenated to create the 10-bit command represent the number of *Shift-DR* states between the *Capture-DR* and *Update-DR* states of CP1 and CP2. The count within CP1 creates the MSBs of the command while the count within CP2 creates the LSBs of the command. The command is decoded when the *Update-DR* state of the second DR-Scan is reached. A determination as to whether the command is a two- or three-part command is made at this point. The command format is shown in *Figure 1048*.

Figure 1048. DR_Scan sequence used for command creation



The 10-bit command created by CP1 and CP2 determines the command's function and whether the command is a two- or three-part command. If the command is a two-part command, the function specified by the command is performed when command part 2 completes. If the command is a three-part command, the DR-Scan following CP2 performs a TAP.7 controller function designated by the 10-bit controller command. This DR-Scan is called a ***Control-Register Scan*** (CR-Scan) and can be any length. A CR-Scan has many of the attributes of the DR-Scan as it moves data between the external tool and TAP.7 controller and accesses various EPU scan paths. There are many two-part commands, but only three three-part commands, each having a special purpose.

60.5.3.4 Commands

There are 11 mandatory and 5 optional commands that can be grouped in classes as shown in the following list:

- Store – The operand is stored in a register or causes an action.
- Select – Controls participation of Scan and conditional command execution.
- Scan – Inputs and outputs with a CR-Scan.
- Enumerate – Controller ID allocation and de-allocation.
- Private – Commands available for chip-specific definition.

The definition of the commands with their operands is shown in *Table 999*.

Table 999. TAP.7 controller command list

Command class	Op-code count	Mnemonic	Name	TAP.7 class			
				1	2	3	4
Store	0x00	STMC	Store Miscellaneous Control	M	M	M	M
	0x01	STC1	Store Conditional 1-bit	M	M	M	M
	0x02	STC2	Store Conditional 2-bit	M	M	M	M
	0x03	STFMT	Store Format	—	M	M	M
	0x04	STDCST	Store Data Channel State	—	—	—	—
Select	0x06	MCM	Make Cond. Group Member	M	M	M	M
	0x07	MSC	Make Scan Group Candidate	—	M	M	M
Scan	0x08	SCNB	Scan Bit	M	M	M	M
	0x09	SCNS	Scan String	M	M	M	M
Enumerate(1)	0x0A	CIDA	Allocate Controller ID	—	—	M	M
	0x0B	CIDD	De-allocate Controller ID	—	—	M	M
Reserved	0x0C–0x1F	Reserved	Reserved	—	—	—	—

1. Enumerate commands are not currently supported by the CJTAG module.

60.5.3.4.1 Store Commands

Store commands are two-part commands that are available in both Star and Series Scan Topologies. The Store Miscellaneous Control (STMC) Command performs control functions that affect all TAP.7 controllers sharing a external tool connection. It stores values in one-bit and two-bit registers.

The Store Conditional 1 bit (STC1) and Store Conditional 2 bit (STC2) Commands store one-bit and two-bit values in registers, respectively. These commands operate in two ways:

1. Unconditionally storing registers in all TAP.7 controllers or
2. Storing registers in only the TAP.7 controllers that have a logic 1 Conditional Group Member (CGM) Register value.

The Store Format (STFMT) Command unconditionally stores a five-bit Scan Format value.

The Store Data Channel State (STDCST) Command stores a five-bit value identifying the states supporting transport.

60.5.3.4.2 Select Commands

Select commands are two-part commands that are available in both Star and Series Scan Topologies. Their primary use is to select TAP.7 controllers of interest in a Star Scan Topology. These commands define the TAP.7 controller(s) participating in scans and the TAP.7 controllers that conditionally execute STC1 and STC2 Commands.

The Make Conditional Group Member (MCM) Command specifies which TAP.7 controllers execute store conditional commands as it manages the Command and Path Enable (CAPE) Register. This register also affects the execution of private commands and is utilized in EPU scan path selection.

The Make Scan Group Candidate (MSC) command manages the SGC register. This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used. Select commands execute only when the CIDI Register is a logic 0, indicating the TAP.7 controller's CID is valid, and are treated as no operations otherwise.

60.5.3.4.3 Scan Commands

Scan commands are three-part commands that are available in both Star and Series Scan Topologies. The CR-Scan portion of Scan Bit (SCNB) and Scan String (SCNS) Commands is used to set and test bits (SCNB) or transfer strings of bits (SCNS). The commands may be used to access both public and private registers.

60.5.3.5 Command Definitions

60.5.3.5.1 Store Commands

60.5.3.5.1.1 Store Miscellaneous Control Command (STMC)

Table 1000. Store Miscellaneous Control

STMC		Store Miscellaneous Control				
Opcode	**Operand**	**bbb**		**x**	**y**	**Description**
00000	bbb x y	000	StateCtl	0	0	NOP
				0	1	ExitCmdLev(ECL)
				1	0	Exit/Suspend (SUSPEND) = 1
				1	1	ZBS Inhibit (ZBSINH) = 1
		001	ScanCtl	0	—	SGC = y
				1	—	CGM = y
		010	RdyCtl	—	—	RDYC = xy
		011	DlyCtl	—	—	DLYC = xy
		100	ScnFunc	0	—	SSDE = y
				1	0	Star-4 Topology Test
				1	1	Reserved
		101	Reserved	—	—	Reserved
		110		—	—	
		111		—	—	

60.5.3.5.1.2 Store Conditional 1-bit (STC1)

Table 1001. Store Conditional 1 bit (STC1)

STC1		Store Conditional 1 bit			
Opcode	Operand	c	bbb	CAPE	Description
00001	c bbb v	0	000	—	SREDGE = v
			001	—	Reserved
			010	—	TRESET = v
			011 – 111	—	Reserved
		1	—	0	No change
			000	1	SREDGE = v
			001	1	Reserved
			010	1	TRESET = v
			011 – 111	1	Reserved

60.5.3.5.1.3 Store Conditional 2-bit (STC2)

Table 1002. Store Conditional 2 bit (STC2)

STC2		Store Conditional 2 bit			
Opcode	Operand	c	bb	CAPE	Description
00010	c bb vv	0	00	—	Reserved
			01	—	STCKDC = vv
			10	—	APFC = vv
			11	—	Reserved
		1	—	0	No change
			00	1	Reserved
			01	1	STCKDC = vv
			10	1	APFC = vv
			11	1	Reserved

60.5.3.5.1.4 Store Scan Format (STFMT)

Table 1003. Store Scan Format (STFMT)

STFMT		Store Scan Format
Opcode	Operand	
00011	nnnnn	SCNFMT = nnnnn

60.5.3.5.2 Select Commands

60.5.3.5.2.1 Make Conditional Group Member (MCM)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the CGM bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The CGM Register is also stored with the STMC and SCNB commands.

Table 1004. Make Conditional Group Member (MCM)

MCM		Conditional Group Member	
Opcode	**Operand**	**m**	**Description**
00110	m iiii	0	CGM bit of the non-targeted controller is cleared
		1	CGM bit of the targeted controller is set. CGM bit of a non-targeted controller is unaffected.

60.5.3.5.2.2 Make Scan Group Candidate (MSC)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the SGC bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The SGC Register is also stored with the STMC and SCNB commands.

Table 1005. Make Scan Group Candidate (MSC)

MSC		Make Scan Group Candidate	
Opcode	**Operand**	**m**	**Description**
00111	m iiii	0	SGC bit of the non-targeted controller is cleared
		1	SGC bit of the targeted controller is set. SGC bit of a non-targeted controller is unaffected.

60.5.3.5.3 Scan Commands

60.5.3.5.3.1 Scan Bit (SCNB)

This is a 3-part command. The operand chooses the bit to be read or written by the CR-Scan.

Table 1006. Scan Bit (SCNB)

SCNB			Scan Bit		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01000	yyyyy	—	0	Scan Group Candidate (SGC)	Write only
		—	1	Conditional Group Member (CGM)	Write only
		—	2	CNFG[0]	Read only
		—	3	CNFG[1] (not implemented)	Read only
		—	4	CNFG[2] (not implemented)	Read only
		—	5	CNFG[3] (not implemented)	Read only
		—	6	Functional reset requested (FRESETR) (not implemented)	Read only
		—	7	Series topology detect	Write only
		—	15 – 8	Reserved	Read only

60.5.3.5.3.2 Scan String (SCNS)

This is a 3-part command. The operand chooses the string to be read or written by the CR-Scan.

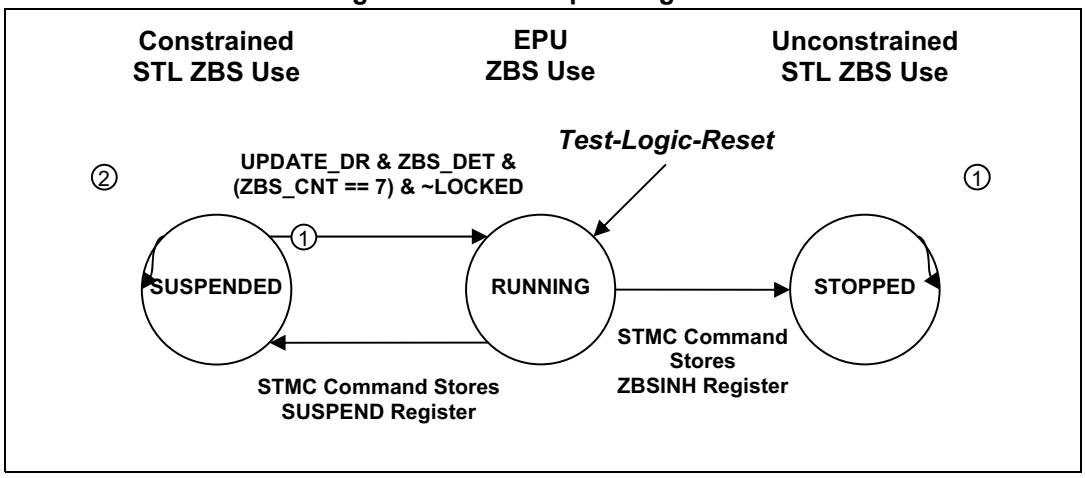
Table 1007. Scan String (SCNS)

SCNS			Scan String		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01001	yyyyy	—	0	RDBACK[0] Register [31:0]	Read only
		—	1	RDBACK[1] Register [31:0] (not implemented)	Read only
		—	2	CNFG[0] Register [31:0]	Read only
		—	3	CNFG[1] Register [31:0] (not implemented)	Read only
		—	4	CNFG[2] Register [31:0] (not implemented)	Read only
		—	5	CNFG[3] Register [31:0] (not implemented)	Read only
		—	6–7	Reserved (read logic 0)	Read only
		—	31–8	Private – available for customization	

60.5.4 EPU operating states

The TAP.7 controller and STL share the use of ZBSs. The EPU has three operating states as shown in *Figure 1049*.

Figure 1049. EPU operating states



The values of the SUSPEND and ZBSINH Registers determine the type of operation as shown in *Table 1008*. The Test-Logic-Reset state sets the SUSPEND and ZBSINH Register values to a logic 0.

Table 1008. ZBS use state characteristics with both TAP.7 and other ZBS use

ZBSINH	SUSPEND	EPU operating state	ZBS use by:	Until
0	0	RUNNING	EPU	Command allocates to STL
0	1	SUSPENDED	STL	ZBS sequence allocates to EPU
1	0	STOPPED	STL	Test-Logic-Reset state allocates to EPU
1	1	Not possible		

60.5.5 System and EPU Paths

The System path is utilized when the external tool requires access to system (STL) resources. This path is used when either the SUSPEND Register is a logic 1 or the ZBS count is less than or equal to one. The EPU path is used otherwise.

The EPU scan path is constructed from five types of EPU scan paths:

- **Bypass** – A 1-bit scan path
- **Bit** – Supports set and test bit operations (a 1-bit shift register)
- **String** – Support 32-bit reads or 32-bit writes
- **Enumerate** – A 36-bit scan path that transports a TAPC address with T3 and above TAP.7s
- **Auxiliary** – An n-bit scan path that transports 1 – n bits using a conventional IEEE 1149.1 style DR-Scan path using Control Levels 4 and 5.

60.6 APU (Advanced Protocol Unit) Operation

60.6.1 Overview

The Advanced Protocol Unit (APU) is placed between the EPU and the RSU to create the T4 TAP.7 controller. The APU delivers advanced capabilities with a number of operating modes (new scan formats) that utilize the Standard Protocol and the Advanced Protocol in both the narrow and wide T4 TAP.7 versions.

The APU enables scan transfers with two signals by serializing the information related to a TAPC state machine state. It provides a number of serialization formats to balance scan flexibility and scan performance. It multiplexes the use of the TMSC signal for T4 TAP.7 controller functions. The serialization is bypassed to provide T3 TAP.7 capability.

With a wide TAP.7, the chip architect may choose to implement the TDIC and TDOC signal functions as fixed functionality (the T3 TDIC and TDOC functions) or as programmable between the T3 TDIC and TDOC signal functions and an alternate function using a TAP.7 controller register. In this case, the default signal function may be either of the programmable choices. Programmable TDIC and TDOC functionality provides a means to fully utilize these signals in any scan topology as the T3 TAP.7 functionality is fully supported.

The entire EPU infrastructure (i.e., ZBS detection, control-level management, and command generation) is utilized when either the Standard Protocol or the Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state machine

state progression. The EPU is unaware of the APU's existence. All T4 features are managed outside of the EPU.

The TAP.7 serialization schemes are specified with scan formats that complement the JScan 0-3 Scan Formats inherited from the T3 TAP.7. The scan formats are called advanced scan formats as they support two-signal operation.

Three groups of scan formats (MScan, OScan0-7, and SScan0-3) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. The OScan and SScan Scan Formats are broken into two groups with the same function, one group has better performance but may only be used with an external tool sourced TCKC. The external tool chooses a scan format that matches the constraints imposed by the application, chip components, and possibly the external tool itself.

The MScan and OScan formats address Test and Debug use cases. The mandatory MScan and OScan0-1 formats emphasize flexibility over performance. The remaining optional Oscan2-7 formats emphasize performance over flexibility by not transmitting unneeded information such as TDI and TDO in non-shift TAPC state machine states.

The optional capabilities added with T4 and above TAP.7s change the bit sequences seen at the TAP.7 signals. A TAP.7 controller comprehends only the bit sequences generated by the use of the features it supports. It is never exposed to bit sequences it does not comprehend. A check for supporting an enabled feature is made before the feature is used (alters the bit sequences). When a T4 TAP.7 controller detects the use of an unsupported scan format, it halts its operation and places itself in a state where its operation may be resumed. This state is henceforth called "offline". A TAP.7 controller that is not offline is henceforth called "online". While offline the TAP.7 controller awaits the detection of a special signaling sequence called the Online Activation Protocol. This protocol sequence places an offline TAP.7 controller online and synchronizes the operation of all online TAP.7s.

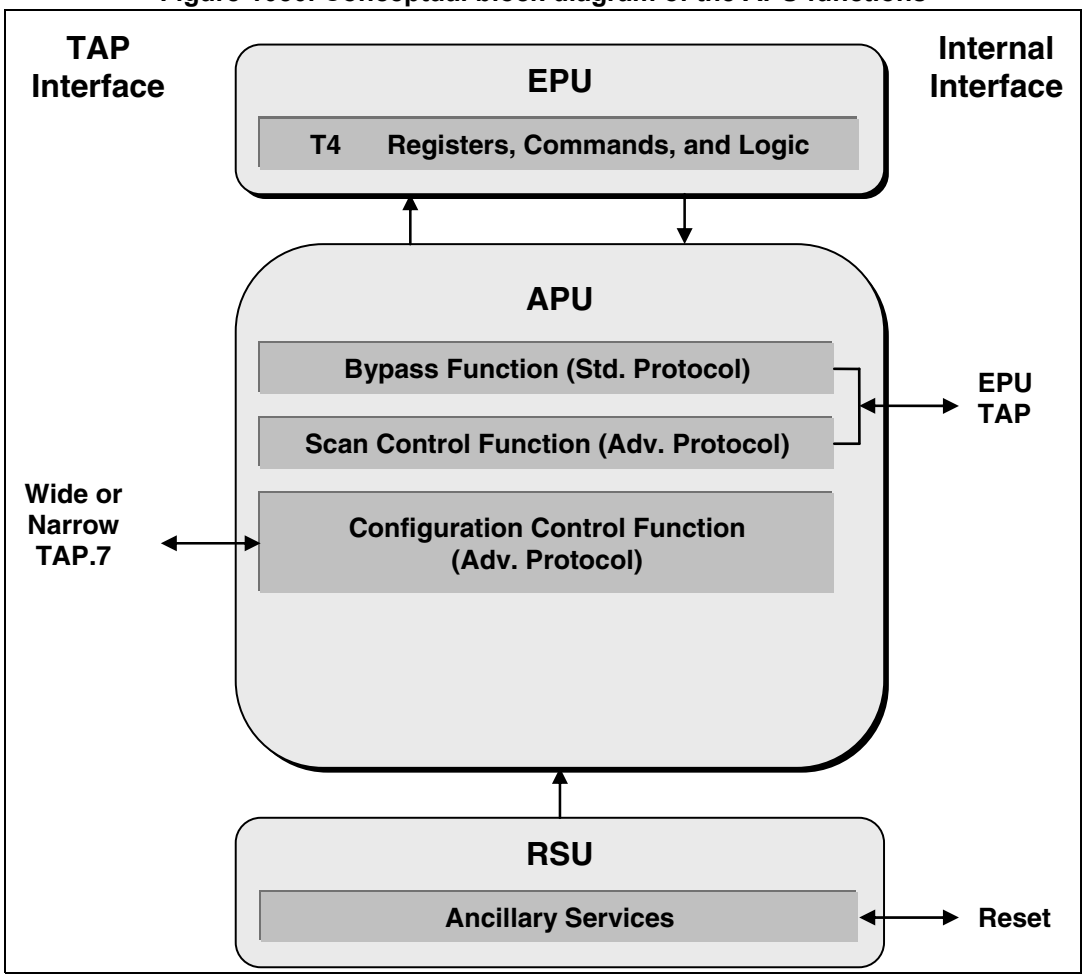
The Advanced Protocol Unit supports the stall of a transfer by both the external tool and STL at any TAPC state.

The APU provides the following functions:

- Bypass (for operation with the Standard Protocol)
- Scan control (for SP use)
- Configuration Control (for CP use)

The Scan Control, Configuration Control, and Transport Control functions are idled when the Standard Protocol is used as the Bypass Function connects the TCKC and TMSC pins to the EPU.

Figure 1050. Conceptual block diagram of the APU functions



The APU's internal interface includes a connection to EPU registers, the EPU's TAP interface, connections to chip-level data channels (when they are implemented), and reset signaling with the EPU. The APU's TAP interface provides mutually exclusive use of the TMSC pin for:

- Bypass Function
- Scan Control Function
- Configuration Control Function

When the Standard Protocol is used, the APU's Bypass function connects the EPU directly to the TAP.7 pins. When the TDIC and TDOC pin functions are not available (narrow TAP.7 or TDIC and TDOC pin functions are an alternate function), the APU sources a logic 1 to the EPU inputs associated with these pins.

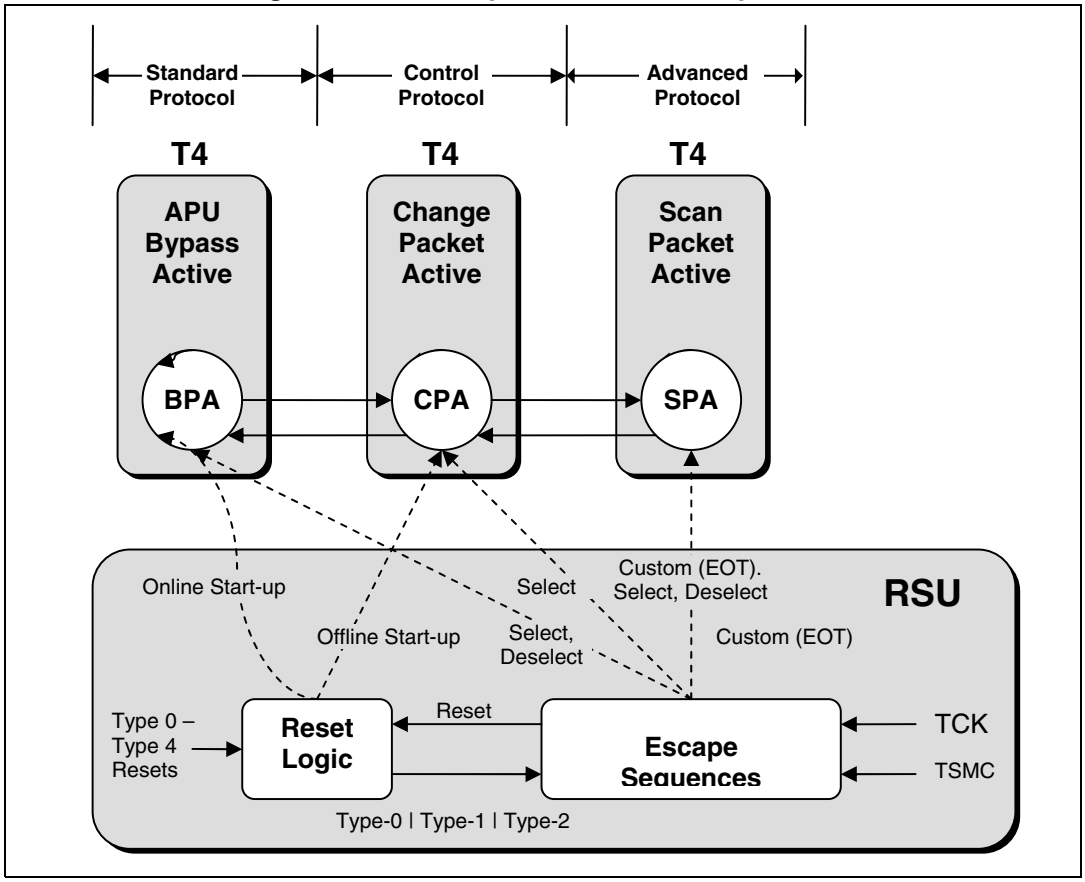
When the Advanced Protocol is used, the APU multiplexes the use of the TMSC pin between the other APU functions. Scan Control acts as a serial to parallel converter, translating the parallel information at the EPU's TAP interface into bidirectional serial transfers. Since the Scan Control is literally a protocol adaptor, the entire EPU infrastructure

(ZBS detection, control-level management, and command generation) is utilized when either the Standard or Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state progression.

60.6.2 Operation

The conceptual view of the APU operation is shown in *Figure 1051*.

Figure 1051. Conceptual view of APU operation



The APU operating states managing the use of the TAP pins and the ancillary (RSU) services use of the TMSC pin.

APU operating states:

- ***BPA*** – **B**y**P**ass **A**ctive
- ***CPA*** – **C**hange **P**acket **A**ctive
- ***SPA*** – **S**can **P**acket **A**ctive

60.6.2.1 BPA

The *BPA* state allocates the use of the TMSC pin to the Standard Protocol. This state is entered two ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up online.
- From the CPA state when a TAP.7 controller register write specifies the use of the Standard Protocol.

60.6.2.2 CPA

The *CPA* state allocates the use of the TMSC pin to a CP. This state is entered in one of three ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up offline.
- A value specifying the use of the Advanced Protocol is written to the SCNFMT Register when using the Standard Protocol.
- A TAP.7 controller register write occurs when using the Advanced Protocol.

60.6.2.3 SPA

The SPA state allocates the use of the TMSC pin to an SP. This state is entered when the SCNFMT Register specifies the use of the Advance Protocol when a CP ends.

60.6.3 Escape sequences

Escape-sequence detection provides services to the Reset Logic, Scan, Configuration Control, and Transport Control functions as outlined below:

- **Scan and Transport Control** – End of Transfer (EOT) escape sequence detected.
- **Configuration Control** – Synchronize Advanced Protocol (SAP) escape sequence.
- **Reset Logic** – Reset (RES) escape sequence detected.

An EOT escape sequence is used to increase the efficiency of OScan Scan formats.

An SAP escape sequence is used to place TAP.7 controllers online.

A RES escape sequence asynchronously causes generation of a Type-3 Reset. Following a RES escape sequence, the drive of the TMSC pin is inhibited so it is not driven when TCKC becomes a logic 0 immediately following the RES escape sequence.

60.6.4 Signal behaviors

The TCKC signal:

- The test clock

The TMSC signal:

- An input when the Standard Protocol is used
- Bi-directional when the Advanced Protocol is used

The TDI and TDO signals:

- Deleted with a narrow TAP.7
- One of two functions with a wide TAP.7
  - Fixed – as the TAP.7 TDIC and TDOC pin functions, or
  - Programmable – as the TAP.7 TDIC and TDOC signal functions or alternate functions with a TAP.7 controller reset establishing the default signal function.

60.6.5 APU Functions

60.6.5.1 Bypass Active (BPA) Function

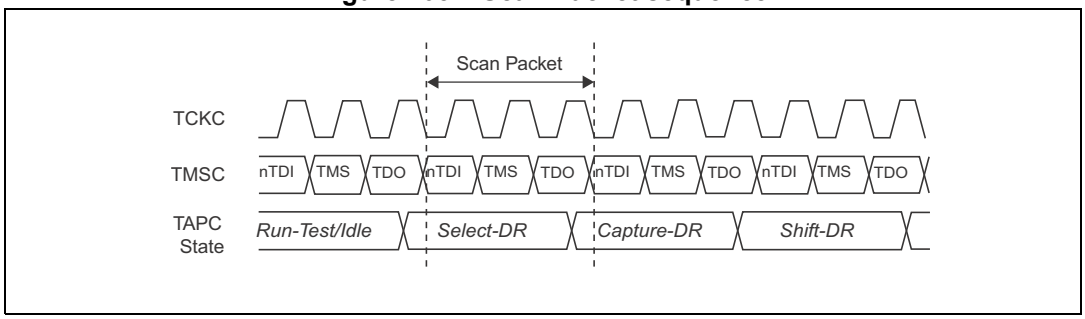
When the Standard Protocol is used, the APU connects the EPU inputs and outputs directly to the TAP.7 pins. If the TDIC and TDOC pin functions are not present (e.g., a narrow TAP.7

or the TDIC and TDOC pins have an alternate function assigned to them), the APU provides a logic 1 to the EPU for unsupported pin functions and ignores the EPU's TDO output data.

60.6.5.2 Scan Packet Active (SPA) Function

The Scan Control function is encapsulated within the Scan Packet Active (SPA) state. With the Standard Protocol, the external tool and TAP.7 exchange the scan information associated with a single TAPC state machine state in one TCK period using the TMS(C), TDI(C), and TDO(C) pins. With the Advanced Protocol all or part of this information is exchanged serially within a Scan Packet (SP) as shown in . There is a one to one correspondence between SPs and TAPC state machine state changes, except when the SPA state precedes the CPA state. In this case the SP preceding a CP does not advance the TAPC state machine state.

Figure 1052. Scan Packet sequence

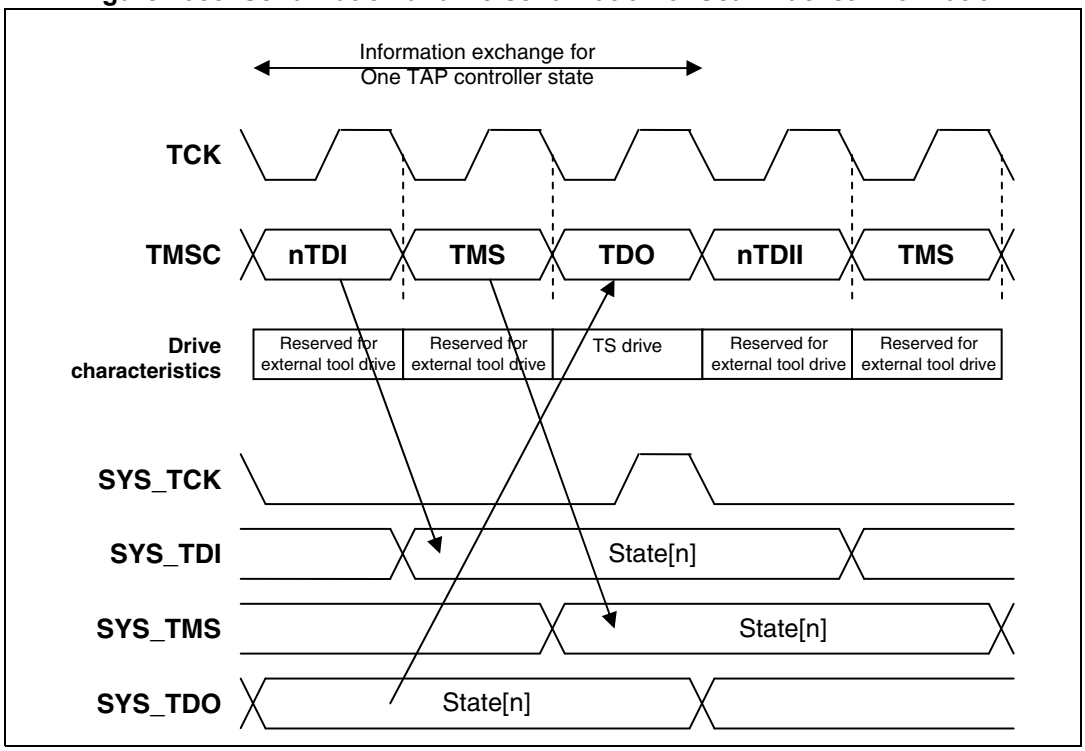


A comparison of the parallel transfer of TAP information with the Standard Protocol and the serialization and de-serialization of the TAP information using the TCKC and TMSC pins with the Advanced Protocol is shown in FIGURE. The IEEE 1149.1 signals between the EPU and STL are prefixed with SYS_.

The APU manages the serialization and de-serialization of the scan information. The APU converts the TDI and TMS information within a Scan Packet (SP) into the TMS and TDI information presented to the EPU. It converts the combination of the EPU TDO and EPU TDO_OE signals into the TDO information in the same SP. If the EPU indicates that the TDO pin drive is not enabled, logic 1 TDO data is created, otherwise the TDO data is the supplied by either the EPU or STL.

Information within a Scan Packet is first passed from the external tool to the TAP.7 followed by information passed from the TAP.7 to the external tool. In this example the scan format specifies the exchange of the TMS, TDI, and TDO TAPC information. Each TAPC state machine state takes three TCKC periods. Other scan formats specify the exchange of different mixes of information. Control information is added with the most flexible scan formats. Some scan formats send less information and consequently take fewer TCKC periods.

Figure 1053. Serialization and De-serialization of Scan Packet Information



In order to produce comparable TAPC state machine state rates with a TAP.1 and TAP.7 controller:

- Full-cycle timing between the external tool and TAP.7 allows operation at higher TCKC frequencies.
- The number of bits transferred to advance a TAPC state machine is minimized. (reaching as few as one for some scan formats.)

The TAP.7 controller provides both full-cycle bit timing (falling TCKC edge to falling TCKC edge) and half-cycle bit timing (falling TCKC edge to rising TCKC edge), with a TAP.7 register used to define the type of timing used. Half-cycle timing is used as the default following a TAP.7 controller reset. The TCKC frequency is reduced to accommodate half-cycle timing.

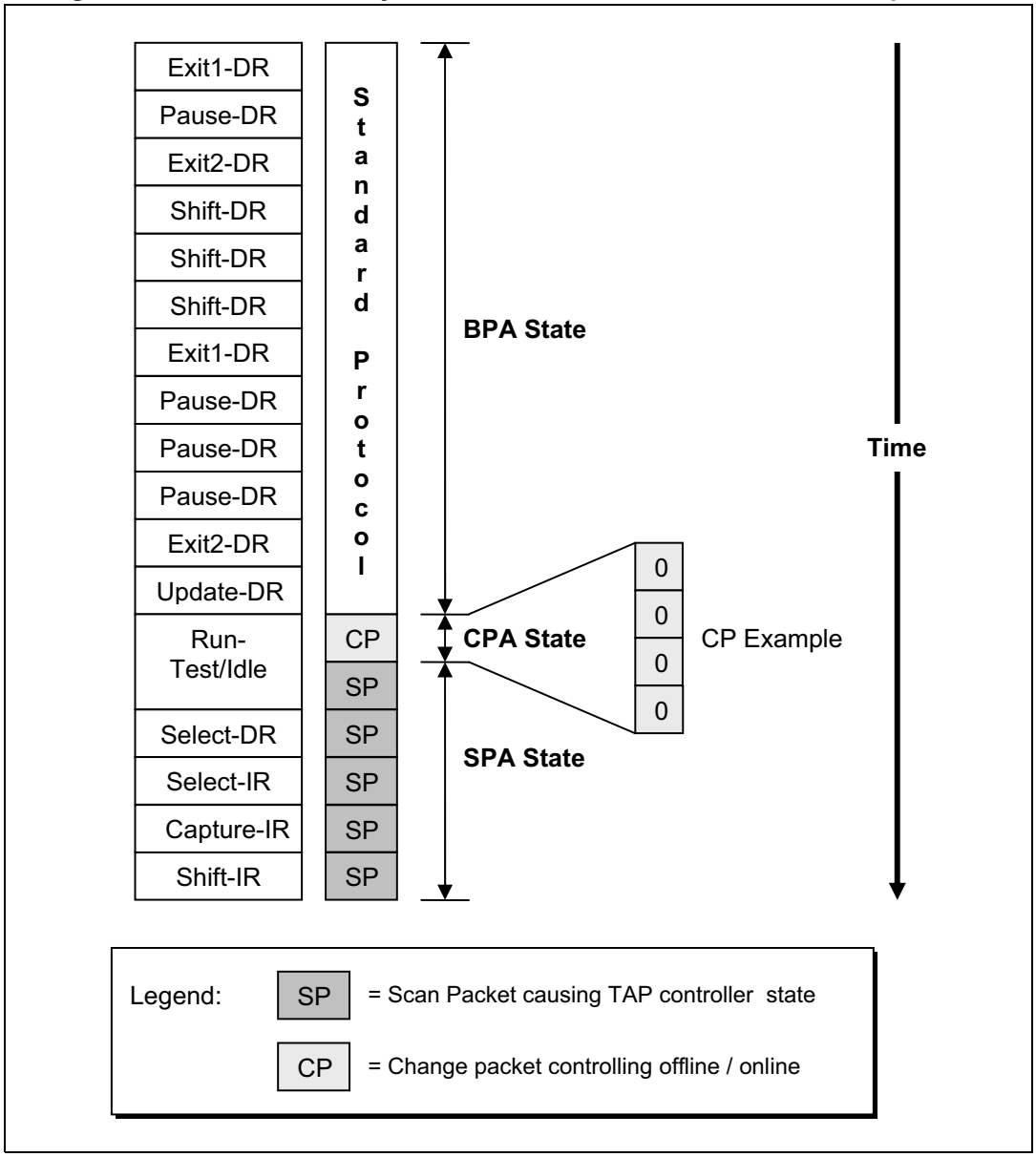
With full-cycle timing, the TMSC input and output data are both output with and sampled with the falling edge of the TCKC. This means that the entire TCKC period may be spent transferring a bit from the external tool to the TAP.7 or vice-versa. The TCKC frequency can therefore be maximized when full-cycle timing is used. With half-cycle timing, the APU samples the TMSC input data with the rising edge of the TCKC and outputs TMSC data with the falling edge of TCKC.

Minimizing the number of bits transferred per TAPC state machine state is affected by a number of factors. The information that must be transferred per TAPC state machine state varies for different test and debug applications. Also, with some test and debug applications, all TAPC state machine states do not require that the same amount of information be transferred. This is especially true with debug applications where specialized use of the TAP is more likely.

60.6.5.3 Control Process Active (CPA) Function

While using the Advanced Protocol, all TAP.7 configuration changes occur within the CPA state. All online and offline transitions also occur while in the CPA state. When entering the CPA state from the BPA or SPA state, the CPA state checks for the use of supported features affecting the Advanced Protocol. The CPA state is exited when the enabled features affecting the Advanced Protocol are supported.

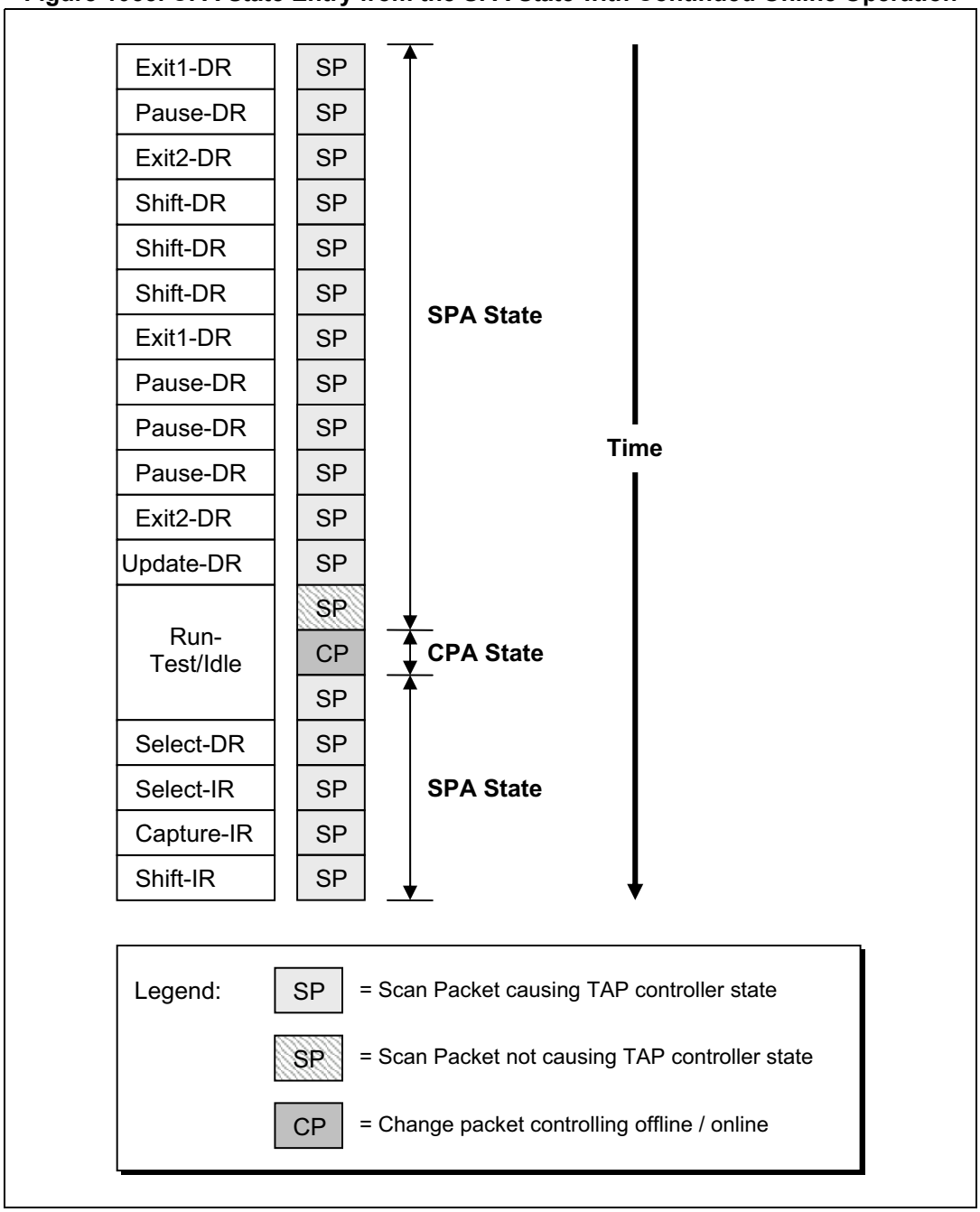
Figure 1054. CPA State Entry from BPA State with Continued Online Operation



An SP preceding a CP does not advance the TAPC state machine state. This SP performs housekeeping functions that assure the contents of the SP preceding this SP have been

fully utilized. This has significance when the SP contains control information that allows the STL to stall the completion of a SP.

Figure 1055. CPA State Entry from the SPA State with Continued Online Operation



When an unsupported feature is enabled, the TAP.7 controller is placed offline. This suspends the TAP.7 controller's operation at a point where its operation may later be synchronized to the remainder of the system. The APU stops advancing the EPU's and CLTAPC's TAPC state machine state while offline. It also suspends Data Channels activity while offline. The detection of the appropriate Online Activation Protocol while the TAP.7 controller is offline places the TAP.7 controller online and causes an exit from the CPA state

to the SPA state. An offline to online transition initiates the use of a predefined set of mandatory features in all online TAP.7 controllers to assure they operate synchronously.

Figure 1056. CPA State Entry from the BPA State with Offline Operation

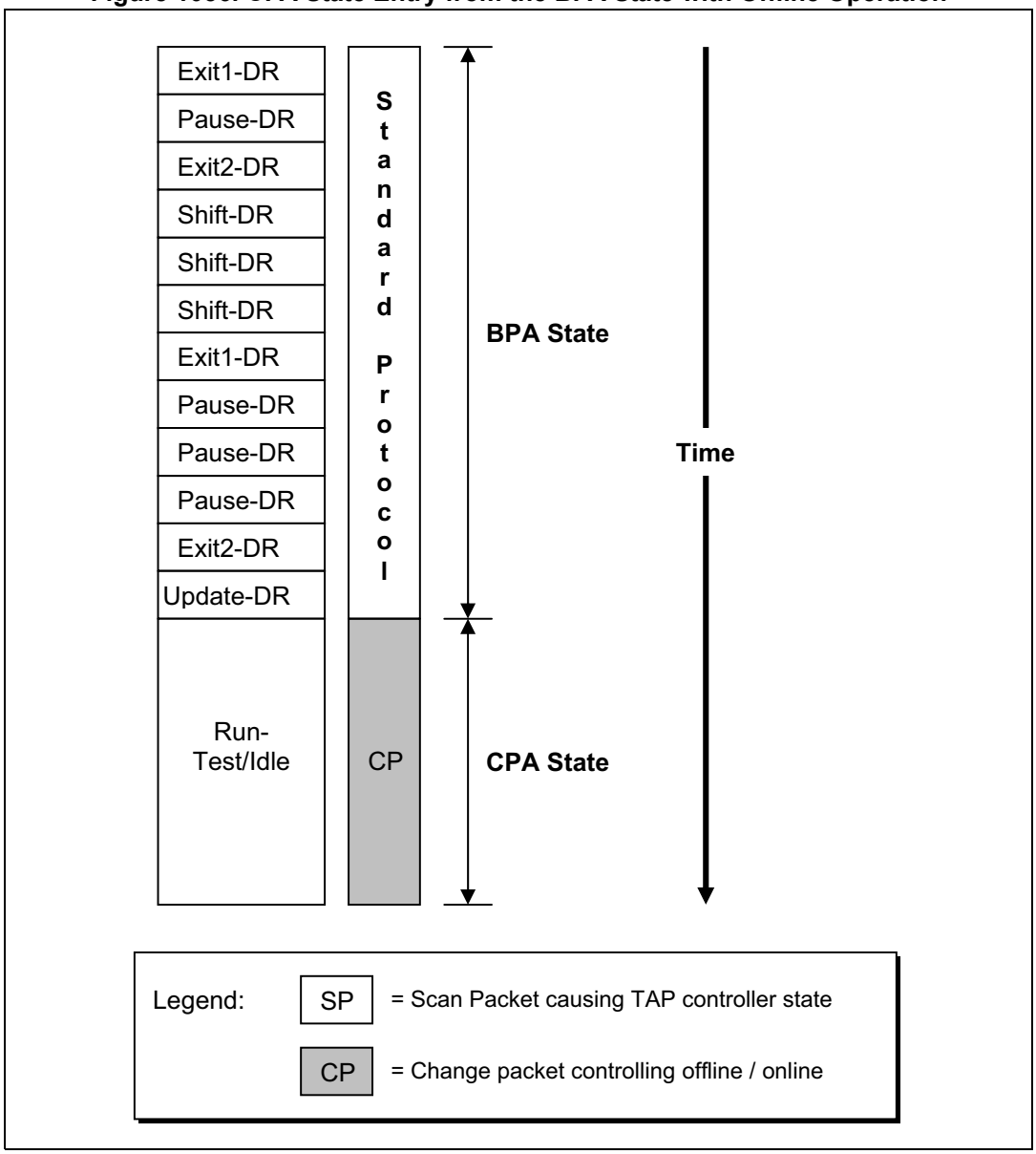
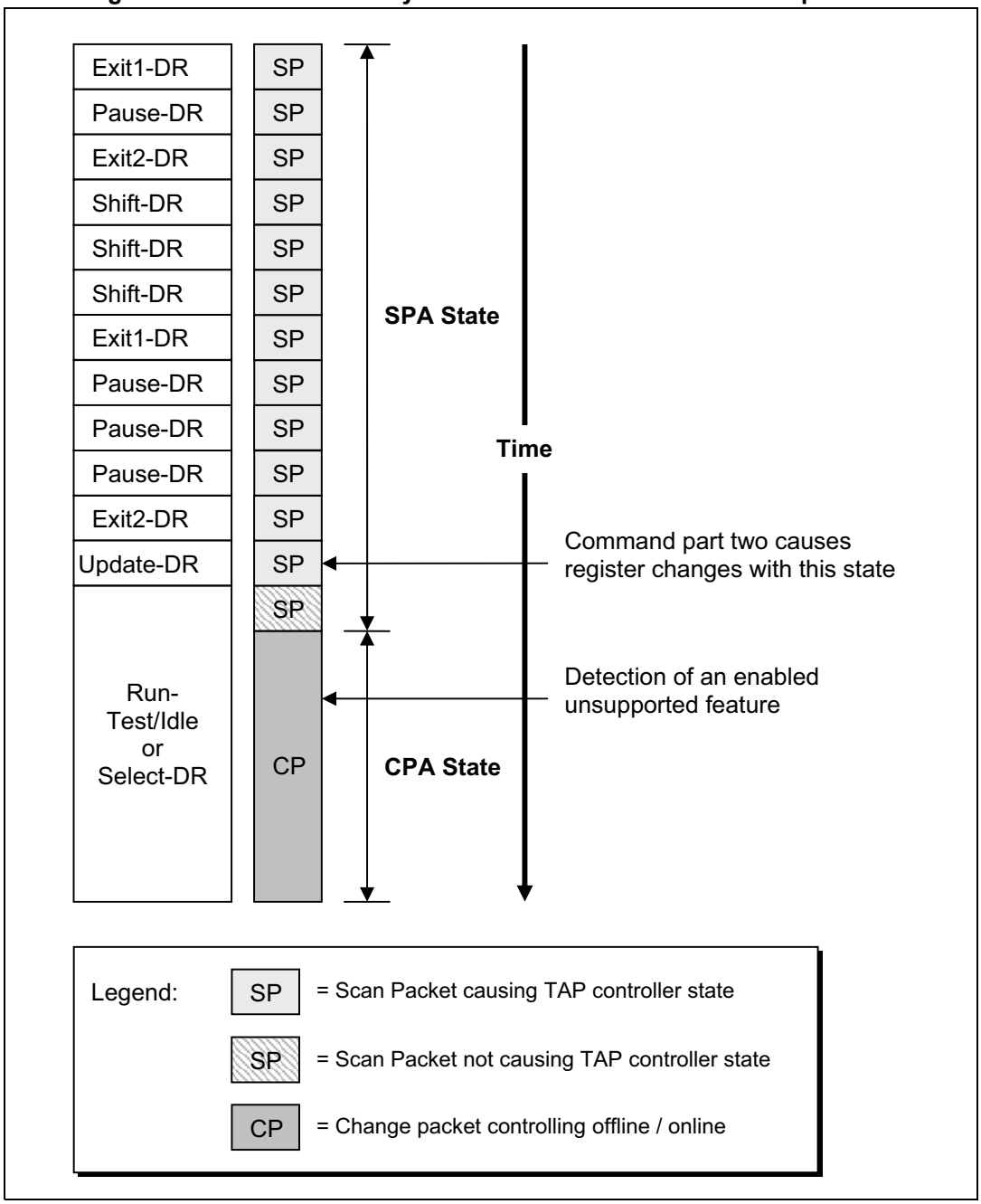


Figure 1057. CPA State Entry from the SPA State with Offline Operation



A TAP.7 controller reset places the TAP.7 controller offline when the offline at start-up option is used. From this point the TAP.7 controller awaits being placed online.

60.6.6 Configuration Change Packets (CP)

A Change Packet (CP) is composed of three element types:

- **Preamble** – A 1-bit value that is a don't care. This bit facilitates a Standard to Advanced Protocol change with some external tool implementations.
- **Body** – Two or more bits representing one or more CP directives
  - **00** – CP_END
  - **01** – CP_NOP (extends the body by 1-bit)
  - **10** – CP_NOP (extends the body by 1-bit)
  - **11** – CP_RES (Generate a Type-3 TAP.7 controller reset coincident with the post amble bit.
- **Post amble** – A 1-bit value that is a don't care. This bit facilitates and Advanced to Standard Protocol change with some external tool implementations.

A minimum-length CP has a Preamble, a 2-bit Body, and a Post amble. This can be as simple as four logic zero values.

60.6.7 Scan Packet

A Scan Packet defines the period of TMSC pin activity where control information exchanged by the external tool and TAP.7 controller affects the TAPC state, except when the SP precedes a Change Packet. The external tool should view the Scan Packet as:

- Exchanging only the necessary TAPC information (TMS, TDI, and TDO or some subset thereof)
- Causing an advance of the TAP.7 TAPC state when it is not followed by a Change Packet.
- Causing an advance of the STL TAPC state when it is not followed by a Change Packet and when the STL is coupled.

With this perspective, the external tool views Scan Packets as running a virtual TAPC state machine at the TAP.7 pins. The effects of each Scan Packet take effect when the Scan Packet completes. The real TAP.7 and STL controller state machine states may change state before or after the virtual state machine.

60.6.8 SP Format

A SP serially exchanges the TDI, TMS, and TDO or a subset thereof, between the external tool and TAP.7 controller. Control information may be added to this exchange. A SP has three parts:

- **Payload** – Data + Control information (all scan formats), the payload content is varied to trade-off performance and flexibility. RDY bit(s) within the payload are included with some scan formats to provide the STL a means to stall progression of the TAPC state.
- **Delay** – Control information (optional) providing separation of adjacent payloads in time. Delay elements support simple external tool implementations where a delay element is used to stall progression of the TAPC state while it develops a new SP following the completion of the prior payload.

The header elements, delay elements, and control information within the payload element is consumed by the TAP.7 controller. This and other control information within the SP payload is not visible to the EPU and STL TAPCs.

60.6.8.1 Payload Element Content

60.6.8.1.1 Input and Output Bit Frames

An input bit frame is sourced by the external tool and an output bit frame is sourced by the TAP.7 controller. The payload may contain and input bit frame, an output bit frame, or both. When an input bet frame arrives at the TAP.7 controller, the TDI and TMS information within it is presented in parallel to the TAP.1 controllers within the EPU and STL. The output bit frame begins when the input frame is completed. The payload is completed when the EPU and STL are ready to utilize this information and the TDO data needed to complete the output bit frame is available. This process is repeated for each SP.

With input bit frames, MScan and OScan scan formats have no control bits input bit frames.

With output bit frames:

- TDO information passed to the external tool is a logic 1 when the EPU indicates the TDO pin would be high impedance.
- RDY bits provide TAP.7 stall opportunities with MScan and some OScan formats.
- PC0 and PC1 bits are added to MScan scan formats to allow Wire-ANDed output.

60.6.8.1.2 Varying the content of input and output bit frames

The various scan formats (MScan, OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. These scan formats vary the content of SPs to balance flexibility and performance for different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. The MScan and OScan scan formats determine the factors affecting the payload content as shown in *Table 1009*.

Table 1009. Factors determining the SP payload content

Scan format	Payload content determined by the:	
	TAPC state	Header
MScan	—	—
OScan	Yes	—

60.6.8.2 Delay Element Content

A delay element can be zero, one, two, or n bits in length. One and two bit delay elements are fixed length with neither the external tool or TAP.7 controller driving the TMSC pin. The TMSC pin is kept at the last value driven by either the external tool or TS. With an n-bit delay elements, the external tool drives the TMSC pin for the first n-1 bits and may drive the TMSC pin the nth bit. With an n-bit delay element the string of delay element bits is a series of directives virtually the same as those used by a CP (DLY_NOP, DLY_END, and DLY_RES).

60.6.8.3 Scan Formats

60.6.8.3.1 Overview

Scan formats utilizing the Advanced Protocol address different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. In one extreme two types of control information, the first allowing the stall of the completion of the Scan Packet, and the second supporting the Wire-AND of TAP.7 controller output are added to the TMS, TDI, and TDO information within the Scan Packet. In the other extreme, the external tool may minimize the information transferred for each TAPC state using its understanding of the application and transfer only the data of interest. In some cases the transfer of only one bit of information per TAPC is required. There are a number of scan formats that operate between these two extremes. The external tool chooses a scan format that best matches the performance/functionality needs of the application.

The scan formats added to specify the use of the Advance Protocol are listed below:

- **MScan** – Maximum flexibility.
- **OScan0-7** – Optimized for test and debug applications.

These scan formats provide seven different functions. Five factors have influenced the definition of these scan formats:

- Supporting the capabilities of the IEEE1149.x and IEEE 1532 Standards.
- Providing the STL and external tool an opportunity to stall the TAPC state progression.
- Providing a minimum pin count and scalable functionality.
- Maximizing the efficiency of scan operations utilized for applications debug operations.
- Providing a rich set of capability for high end applications debug.

The performance and flexibility of MScan and OScan scan formats are shown in *Table 1010* and *Table 1012*, respectively. The performance and flexibility of these scan formats are given subjective/approximate ratings in these tables. These ratings range from least to best. Use cases for these scan formats are also included in these tables.

60.6.8.3.2 MScan scan formats

The MScan scan format provides a universal scan function. It is the most flexible but lowest-performance scan format. The Scan Packets used by these scan formats are packed with data and control information and is the same for all TAPC states. This scan format accommodates the deployment of IEEE 1149.1 IP with non-compliant behavior within the STL. The STL is allowed to stall the TAPC state progression (to pace the arrival of Scan Packets with its ability to process them), and uses Wire-ANDed Output.

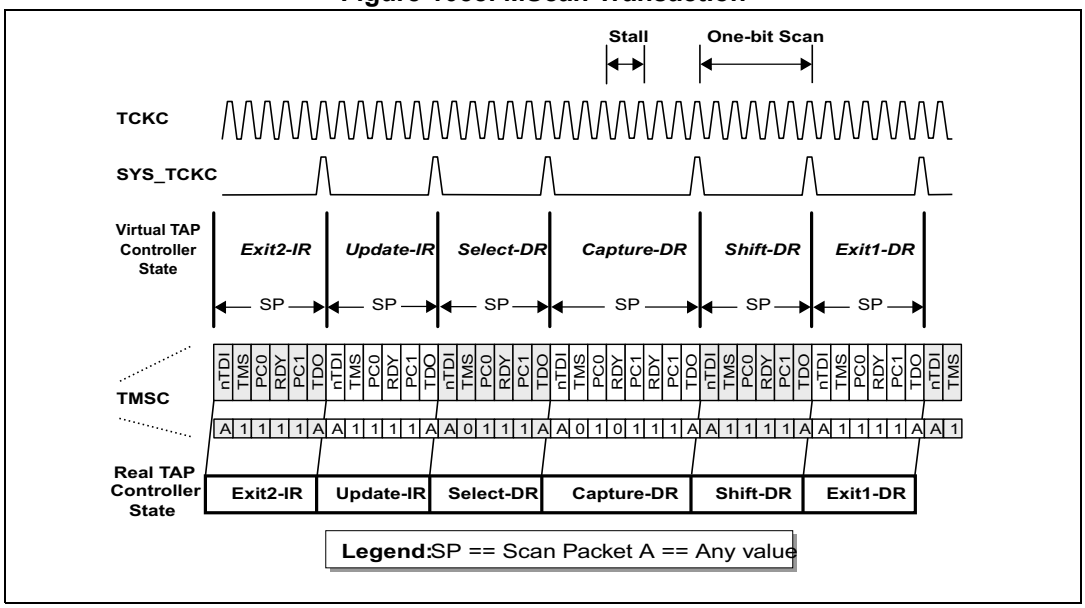
Table 1010. MScan scan format use case summary

Scan format	Supporting	Performance	Flexibility	TCKC source
MScan	STL Stalls	Least	Best	External tool or TS
	Test			
	Non-compliant behavior STL IP			
	Multi-chip debug communication			
	Controller ID Allocation			

Table 1011. MScan Scan Packet content

	Non-shift TAP controller states							Shift TAP controller states					
MScan	nTDI	TMS	PC0	RDY	PC1	TDO		nTDI	TMS	PC0	RDY	PC1	TDO

Figure 1058. MScan Transaction

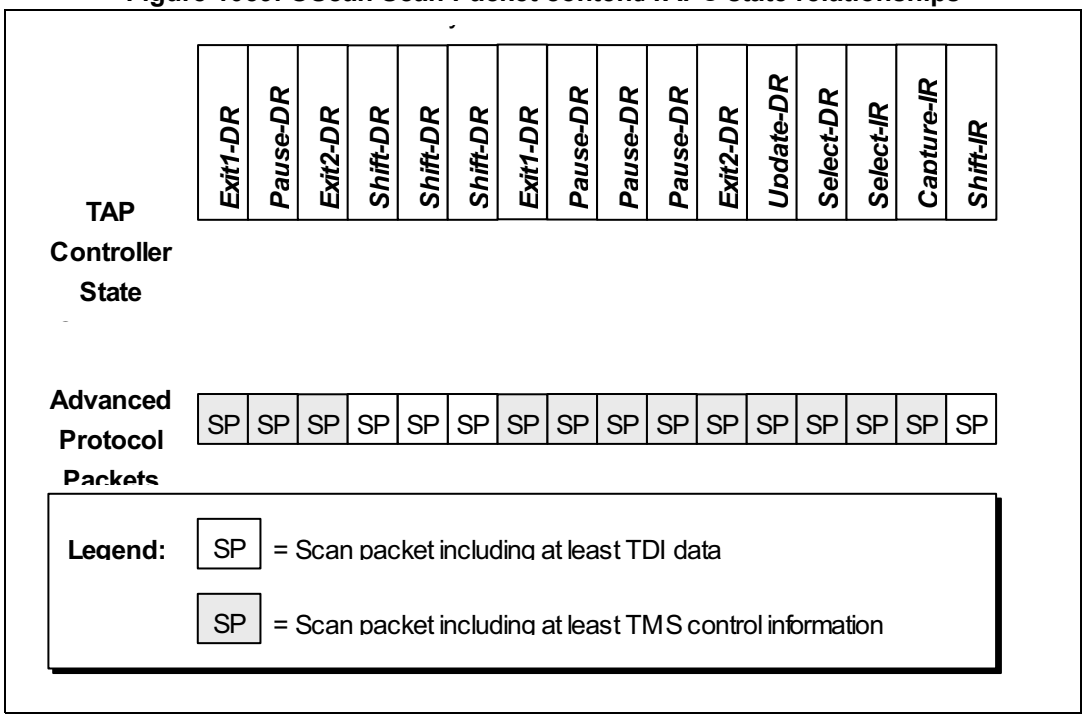


60.6.8.3.3 OScan Scan Formats

The OScan scan formats provide optimization choices where no knowledge of the data content of the scan exchange is needed. The Scan Packet content of these scan formats is modulated by a combination of the scan format and the TAPC state for some OScan scan formats. For instance, TDI data and TDO data is not exchanged for non-shift TAPC states

and included in shift TAPC states in certain OScan formats. The modulation of Scan Packet content with TAPC state varies by scan format.

Figure 1059. OScan Scan Packet content/TAPC state relationships



OScan scan formats replicate four functions in two forms:

- **OScan0 – OScan3:** Providing the best performance for a test clock source by the target system.
- **OScan4 – OScan7:** Providing the best performance for a test clock sourced by the external tool.

The OScan4 – OScan7 scan formats are more efficient, as EOT escape sequences are used to identify an exit from the Shift-xR TAPC state. A Scan Packet associated with the Shift-xR TAPC state does not include a TMS bit. Instead, and EOT escape sequence overlaid on an nTDI bit causes an exit from shift TAPC state with no exit from this state occurring otherwise. A look at the 32-bit DR-Scan shows the impact of this optimization. A total of 96 bits are required when each Scan Packet associated with a Shift TAPC state when TMS is included in these Scan Packets while a total of 64 bits are required when TMS is not included in these Scan Packets. Assuming the EOT escape adds the equivalent of three bit periods, the number of TCK periods to this transfer is 67/96 or roughly 70% of the total without this optimization.

These scan formats may only be used with an external tool sourced TCKC as EOTs are utilized to improve its performance. The OScan0 – OScan3 Scan Formats always have a TMS bit included in the Scan Packet. The OScan scan format use case summary is shown in *Table 1012*.

Table 1012. OScan scan format use case summary

Scan Format	Supporting	Performance	Flexibility	TCKC Source
OScan0	Stalls	↓	Best	External tool
OScan1	Test		↑	
OScan2	Debug Performance			
OScan3	Debug Downloads (input only)	Best		
OScan4	Stalls	↓	Best	External tool or TS
OScan5	Test		↑	
OScan6	Debug Performance			
OScan7	Debug Downloads (input only)	Best		

Certain OScan optimizations are applied in a hierarchical manner.

- **Optimization I:** OScan1/OScan5 – RDY bit(s) and their trailing ONE bits are deleted from all Scan Packets.
- **Optimization II:** OScan2/OScan6 – Optimization I + nTDI and TDO bits are deleted from Scan Packets associated with non-shift TAPC states.
- **Optimization III:** OScan3/OScan7 – Optimization II + TDO bits are deleted from Scan Packets.
- **Optimization IV:** OScan4 – OScan7 – TMS bits are deleted from Scan Packets associated with shift TAPC states.

Table 1013. OScan Scan Format "Payload"

Scan Format	Non-shift TAP controller states					Shift TAP controller states			
OScan0	nTDI	TMS	RDY	TDO		nTDI	TMS	RDY	TDO
OScan1	nTDI	TMS	—	TDO		nTDI	TMS	—	TDO
OScan2	—	TMS	—	—		nTDI	TMS	—	TDO
OScan3	—	TMS	—	—		nTDI	TMS	—	—
OScan4	nTDI	TMS	RDY	TDO		nTDI	—	RDY	TDO
OScan5	nTDI	TMS	—	TDO		nTDI	—	—	TDO
OScan6	—	TMS	—	—	Note 1	nTDI	TMS	—	TDO
					Note 2	nTDI	—	—	TDO
OScan7	—	TMS	—	—		nTDI	—	—	—

Note 1: Following Capture-xR or Exit2-xR Scan Packet

Note 2: Following Shift-xR Scan Packet

60.7 Functional Description

This section combines the information given in the Ancillary Services, EPU Operation, and APU Operation sections into a single reference for performing common TAP.7 tasks.

60.7.1 Switching from Standard Protocol to Advanced Protocol

Upon exit of Type-0 to Type-4 reset, the CJTAG begins operation in Standard Protocol 4-pin mode. The following steps are required to move from Standard Protocol 4-pin mode to Advanced Protocol (either 4-pin or 2-pin) mode.

The first step required to move to the Advanced Protocol is to change the CJTAG control level to control level 2 via zero-bit scans (ZBS), as described in *Section 60.5.1.4, Control levels*.

After the control level is set to control level 2, TAP.7 commands can be executed as described in *Section 60.5.3, EPU Commands*. Execution of the Store Scan Format (STFMT) to set the scan format to any OSCAN0-7 or MSCAN format begins the switch from the Standard Protocol to the Advanced Protocol.

Once the STFMT command is executed selecting an Advanced Protocol scan format, the CJTAG transitions to the intermediate Control Protocol mode. The Control Protocol uses TMS inputs to control Configuration Change Packets as described in *Section 60.6.5.3, Control Process Active (CPA) Function* and *Section 60.6.6, Configuration Change Packets (CP)*. Loading a change packet body value of 00 (CP_END) results in the switch to the Advanced Protocol.

If Advanced Protocol 2-pin operation is desired, the APFC register should be written to a value of 10 or 11 prior to the execution of the STFMT command that selects an advanced scan format. The APFC register is written via the Store Conditional 2-bit (STC2) command described in *Section 60.5.3.5.1.3, Store Conditional 2-bit (STC2)*.

61 JTAG Data Communication (JDC)

61.1 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.

61.2 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.

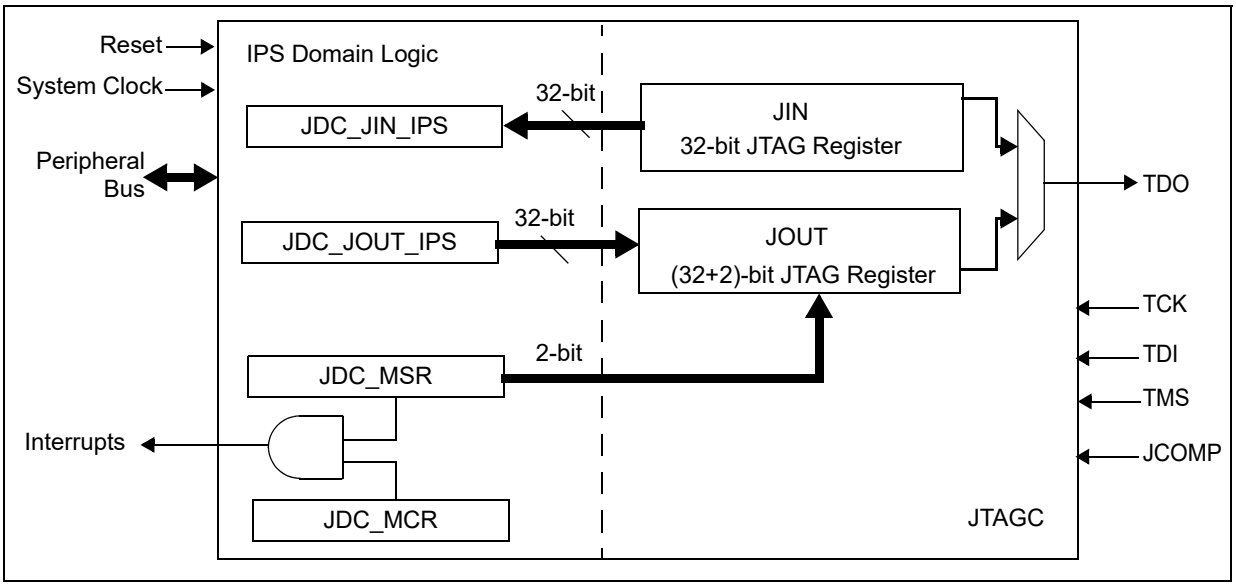
The JDC implements the following IPS data registers, occupying separate memory space:

- 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 1060*.

Figure 1060. JDC block diagram



61.3 Register definition

Table 1014 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 1014. JDC memory map

Address offset (hex)	Register	Access	Reset value	Section/page
0x0	Module Configuration Register (JDC_MCR)	R/W	0x0000_0000	*61.3.1.1/1760*
0x4	Module Status Register (JDC_MSR)	W1C	0x0000_0000	*61.3.1.2/1761*
0x8	JTAG Output Data Register (JDC_JOUT_IPS)	R/W	0x0000_0000	*61.3.1.3/1761*
0xC	JTAG Input Data Register (JDC_JIN_IPS)	Read Only	0x0000_0000	*61.3.1.4/1762*

61.3.1 Register descriptions

61.3.1.1 Module configuration register (JDC_MCR)

Figure 1061 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.

Address: 0x00 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	JIN_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	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	JOUT_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1061. Module Configuration Register (JDC_MCR)

The JDC_MCR is described in *Table 1015*.

Table 1015. JDC_MCR register field descriptions

Field	Description
JIN_IEN	JIN Interrupt Enable. 0 Setting of JDC_MSR[JIN_INT] bit does not assert the JIN interrupt 1 Setting of JDC_MSR[JIN_INT] bit asserts the JIN interrupt
JOUT_IEN	JOUT Interrupt Enable. 0 Setting of JDC_MSR[JOUT_INT] bit does not assert the JOUT interrupt 1 Setting of JDC_MSR[JOUT_INT] bit asserts the JOUT interrupt

61.3.1.2 Module status register (JDC_MSR)

Figure 1062 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.

Address : 0x04 Access: User w1c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	JIN_RDY	0	JIN_INT
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	JOUT_RDY	0	JOUT_INT
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1062. Module Status Register (JDC_MSR)

The JDC_MSR is described in *Table 1016*.

Table 1016. JDC_MSR register field descriptions

Field	Description
JIN_RDY	JIN Ready (read only). 0 Cleared upon software read of JDC_JIN_IPS contents via IPS 1 Set when new data is written to the JDC_JIN_IPS register
JIN_INT	JIN Interrupt. 0 Cleared by writing logic 1 1 Set when new data is written to the JDC_JIN_IPS register
JOUT_RDY	JOUT Ready (read only). 0 Cleared upon tool read of JOUT register via JTAG port 1 Set when new data is written to the JDC_JOUT_IPS register
JOUT_INT	JOUT Interrupt. 0 Cleared by writing logic 1 1 Set when JOUT_RDY bit is cleared by tool reading JOUT register

61.3.1.3 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 1063* shows the format of the JDC_JOUT_IPS register. This register is reset by system destructive reset.

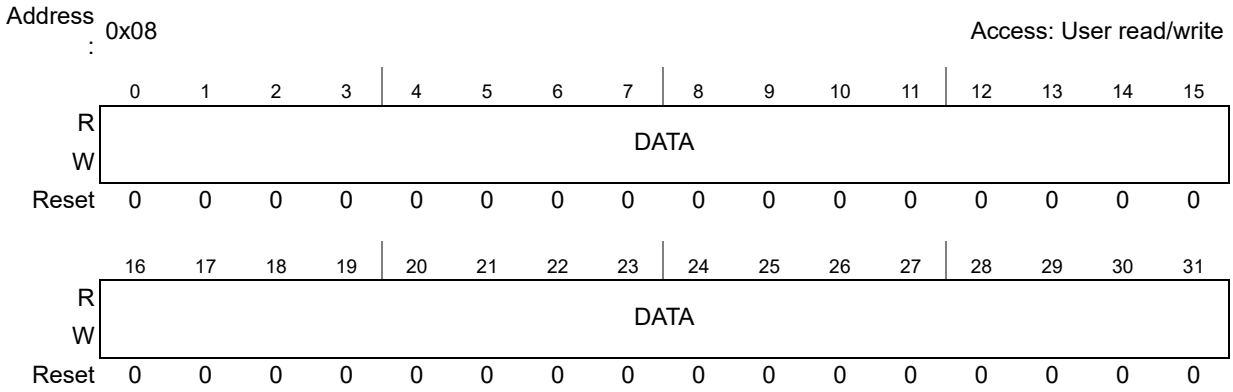


Figure 1063. JDC_JOUT_IPS register

The JDC_JOUT_IPS register is described in *Table 1017*.

Table 1017. JDC_JOUT_IPS register field descriptions

Field	Description
DATA	Contains JOUT_IPS data.

61.3.1.4 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 1064* shows the format of the JDC_JIN_IPS register. This register is reset by system destructive reset.

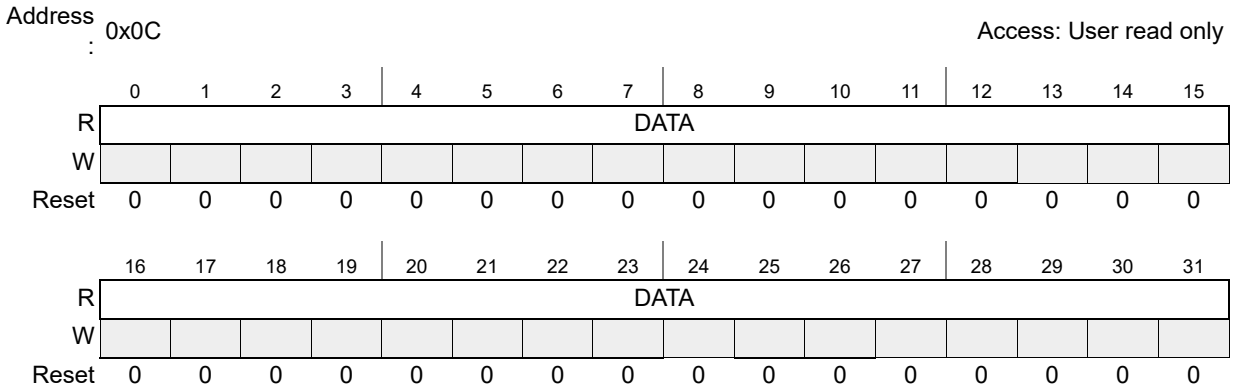


Figure 1064. JDC_JIN_IPS register

The JDC_JIN_IPS register is described in *Table 1018*.

Table 1018. JDC_JIN_IPS register field descriptions

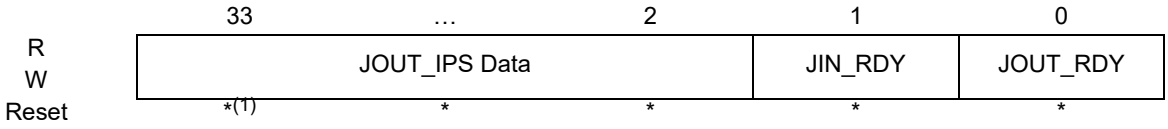
Field	Description
DATA	Contains JIN_IPS data.

61.3.2 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.

61.3.2.1 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 JIN_RDY and JOUT_RDY status bits. *Figure 1065* shows the format of the JOUT register. This register is reset by system destructive reset and JTAG reset.



1. The reset value of the JOUT register is 34'b0.

Figure 1065. JOUT JTAG register

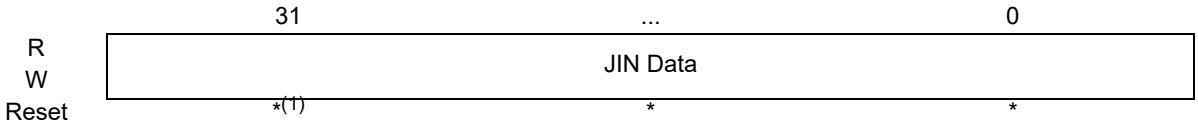
The JOUT register is described in *Table 1019*.

Table 1019. JOUT JTAG register field descriptions

Field	Description
JOUT_IPS Data	Data value from JDC_JOUT_IPS register
JIN_RDY	State of JIN_RDY bit from JDC_MSR
JOUT_RDY	State of JOUT_RDY bit from JDC_MSR

61.3.2.2 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 1066* shows the format of the JIN register. This register is reset by system destructive reset and JTAG reset.



1. The reset value of the JIN register is 32'b0.

Figure 1066. JIN JTAG Register

The JIN register is described in *Table 1020*.

Table 1020. JIN JTAG register field descriptions

Field	Description
JIN Data	Contains data to be captured in JIN_IPS register upon exit of Update-DR state when executing WRITE_JIN JTAG instruction.

61.4 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 JDC_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 JDC_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 JDC_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 JDC_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.

61.4.1 IPS register access

IPS access is available to the registers described in *Section 61.3.1: Register descriptions*.

61.4.2 JTAG register access

Refer to the JTAGC documentation for information on how to access the JTAG registers.

61.4.2.1 JDC block instructions

The JDC block implements the IEEE 1149.1-2001 defined instructions listed in *Table 1021*. 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 1021. JTAG instructions

Instruction	Code[4:0]	Instruction Summary
Reserved	00001	Factory debug reserved
JOUT_READ	00010	Selects JOUT data register. Data from JDC_JOUT_IPS is captured into JOUT data register upon entry to Capture-DR state while JOUT_READ is active.

Table 1021. JTAG instructions(Continued)

Instruction	Code[4:0]	Instruction Summary
JIN_WRITE	01110	Selects JIN data register. Data from JIN is captured into JDC_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

62 JTAG Master (JTAGM)

62.1 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.

62.1.1 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 or from the LFAST through the parallel interface. The JTAGM has the capability to differentiate between data from software and LFAST. 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 to the LFAST through another parallel interface. The JTAGM has the following registers:

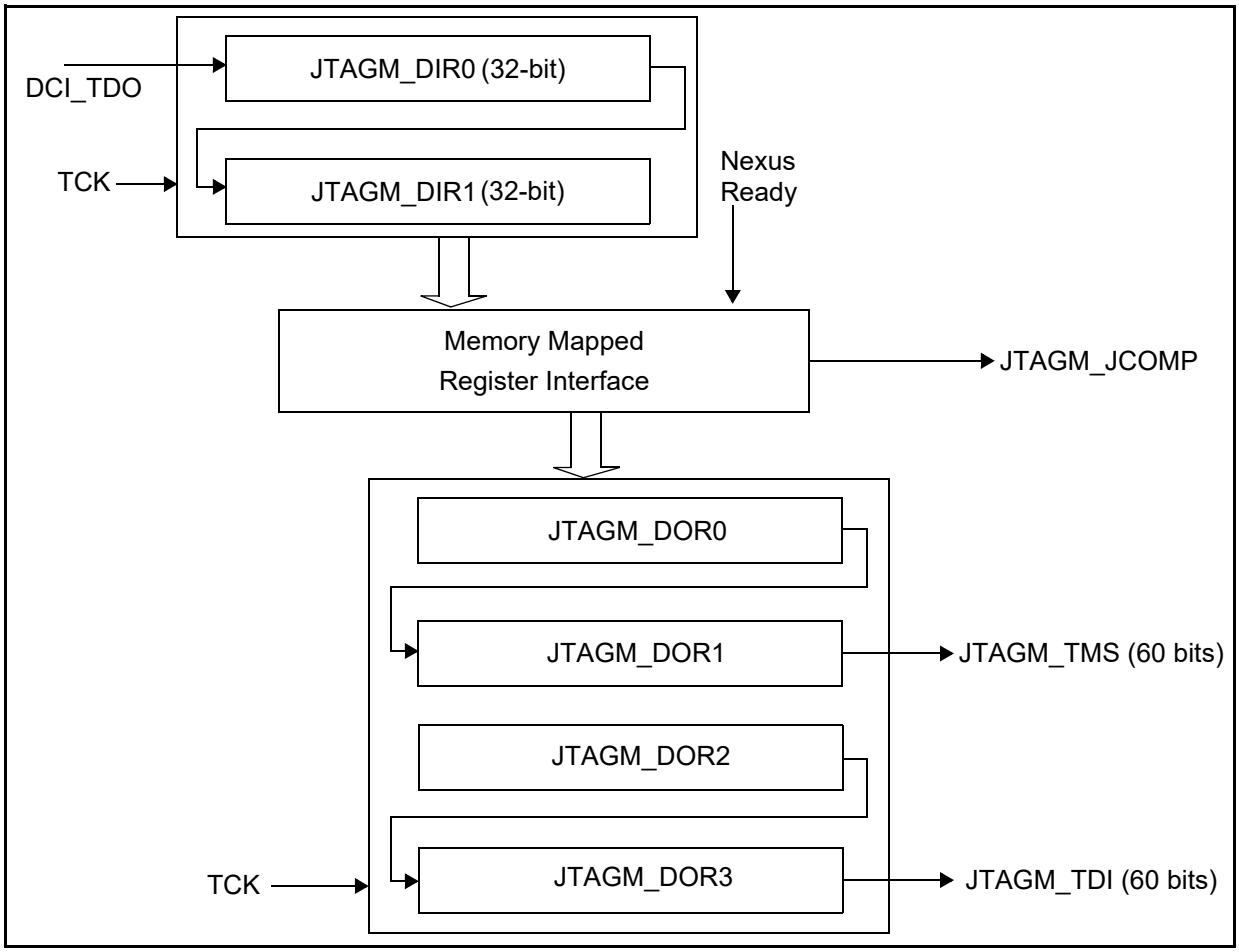
- Configuration register
- Status register
- Four data output registers
- RXCRC receive register
- Two data input registers

All these registers, described in detail in *Section 62.4: Memory map and register definition*, 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 1067 shows the block diagram of the JTAGM.

Figure 1067. JTAGM block diagram



62.1.2 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
- Receives/provides JTAG data from/to the LFAST

62.2 Functional description

62.2.1 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.

62.2.2 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.

62.2.2.1 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.

62.2.2.2 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.

62.2.3 $\overline{\text{EVTO}}$ 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.

62.2.4 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 62.2.2.1: Status register ready bit*). The configuration register is accessed via the parallel input port by a 128-bit payload arriving on channel A. The 128-bit payload is made from the following:

- 32-bit mask of which bits within the configuration register should be written +
- 32-bit value to be written to the configuration register +
- 24-bit mask of which bits within the status register should be written +
- 8 bits of zero-padding +
- 24 bits of data to be written to the status register +
- 8-bit CRC (the least significant 8 bits of the status register cannot be written because they contain automatically generated CRC data).

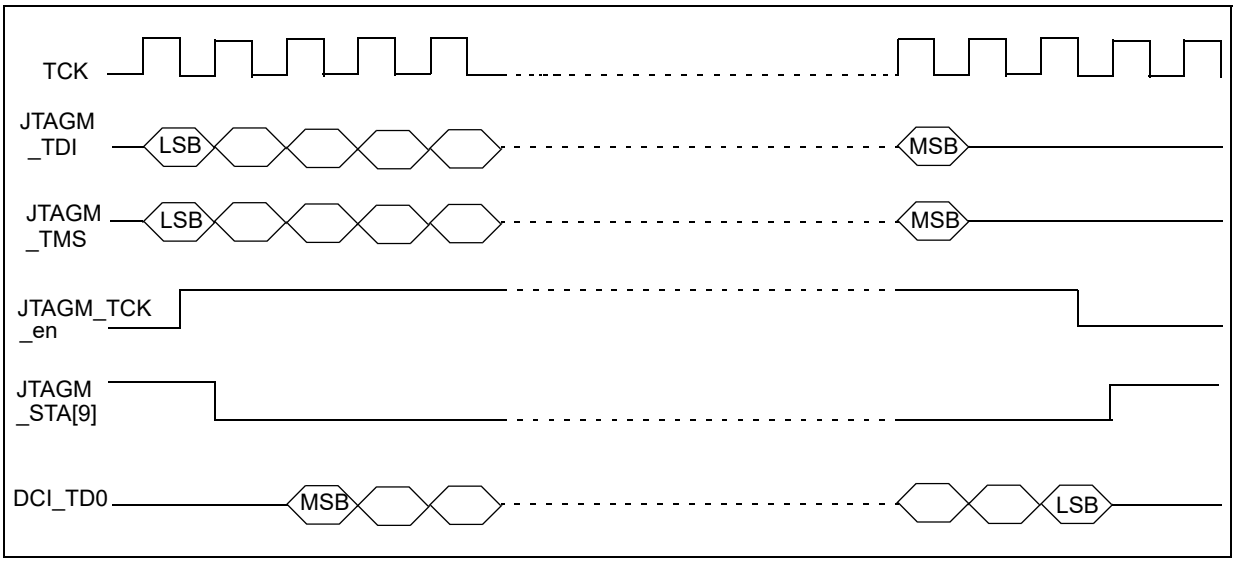
The masked data is transferred to the 32-bit configuration register and 32-bit status register. The JTAGM module responds to this configuration/status write by outputting a 96-bit payload to the parallel output port to channel number A. The 96-bit payload is made up of the 32-bit value of the configuration register plus 32 bits of zero-padding plus 32-bit value of the status register (including the least significant 8 bits which are an 8-bit CRC calculated on the previous 88 bits).

While the current messages are being transmitted from the second buffering (memory mapped registers), the first buffers are continuously filled up. The status register also includes status bits provided by the DCI module.

62.2.5 JTAGM to DCI serial interface

Figure 1068 shows the data transfer timings between JTAGM and DCI.

Figure 1068. JTAGM to DCI serial interface diagram



62.2.6 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.

62.3 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

62.4 Memory map and register definition

Table 1022 contains the memory map of the JTAGM registers.

Table 1022. Memory map

Offset	Register	Location
0x00	Module Configuration Register (JTAGM_MCR)	*on page 1770*
0x04	Status Register (JTAGM_SR)	*on page 1772*
0x08	Data Out Register 0 (JTAGM_DOR0)	*on page 1774*
0x0C	Data Out Register 1 (JTAGM_DOR1)	*on page 1775*
0x10	Data Out Register 2 (JTAGM_DOR2)	*on page 1775*
0x14	Data Out Register 3 (JTAGM_DOR3)	*on page 1775*
0x18	Reserved	
0x1C	Data Input Register 0 (JTAGM_DIR0)	*on page 1776*
0x20	Data Input Register 1 (JTAGM_DIR1)	*on page 1776*

62.4.1 Register descriptions

62.4.1.1 Module configuration register (JTAGM_MCR)

Figure 1069 shows the format of the JTAGM_MCR.

Offset: 0x00 Access: User read/write

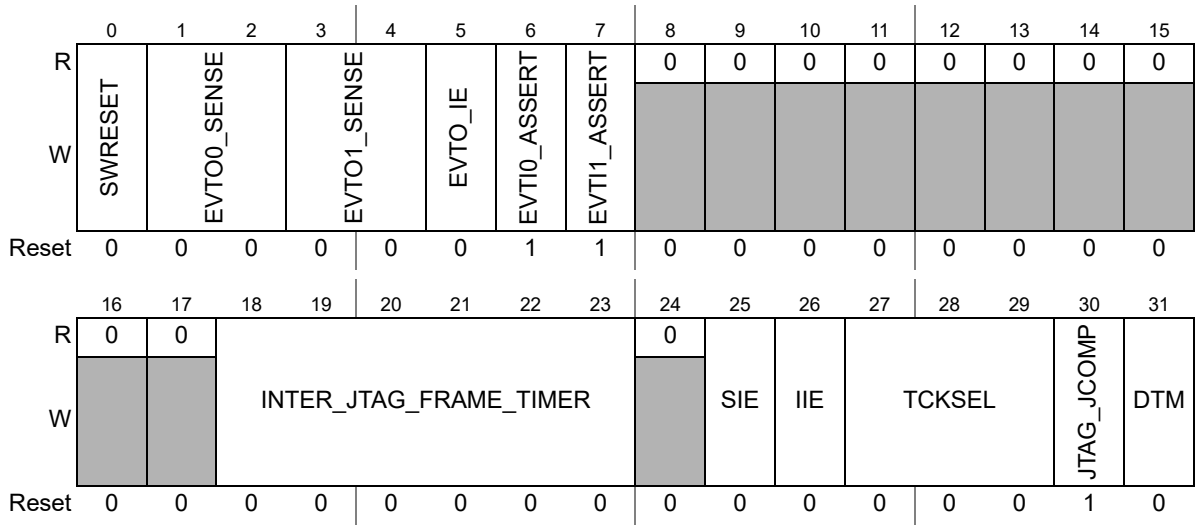


Figure 1069. JTAGM_MCR register

The JTAGM_MCR register is described in *Table 1023*.

Table 1023. 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 Asserted the $\overline{\text{EVTO}}$0 from the JTAGM module 1 Deasserts the $\overline{\text{EVTO}}$0 from the JTAGM module

Table 1023. JTAGM_MCR register field descriptions(Continued)

Field	Description
7 EVTI1_ASSERT	EVTI1 Assert. 0 Asserted the $\overline{\text{EVTO1}}$ from the JTAGM module 1 Deasserts the $\overline{\text{EVTO1}}$ from the JTAGM module **Note:** In the SW mode (when DTM =1), regardless of the state of “EVTI1_ASSERT” bit the internal “evti” signal remains de-asserted . Though this bit can be written , yet writing to this bit has no effect.
8–17	Reserved
18–23 INTER_JTAG_ FRAME_TIMER	TCK delay. 0000000 TCK delay 0000011 TCK delay ... 11111163 TCK delay
24	Reserved
25 SIE	SPU Interrupt Enable. 0 JTAGM does not generate an interrupt to the CPU 1 JTAGM generates an interrupt to the CPU if the SPU interrupt request is asserted
26 IIE	Idle Interrupt Enable. 0 JTAGM does not generate an interrupt to the CPU upon completion of a 60-bit JTAG transfer 1 JTAGM generates an interrupt to the CPU upon completion of a 60-bit JTAG transfer
27–29 TCKSEL	TCK Divider division factor control value. (TCK Divider divides the clock input to JTAGM IP to generate TCK out (input to DCI)). 000 TCK Divider division factor is 1 001 TCK Divider division factor is 2 010 TCK Divider division factor is 3 ... 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 Transfer Mode 1 Data for JTAG transferred by software. 0 Data transferred by software disabled.

62.4.1.2 Status register (JTAGM_SR)

The functionality of some of the SR bits changes depending on the setting of the DTM bit in the MCR. These differences are described in the field descriptions table.

Offset: 0x04 Access: User read/write

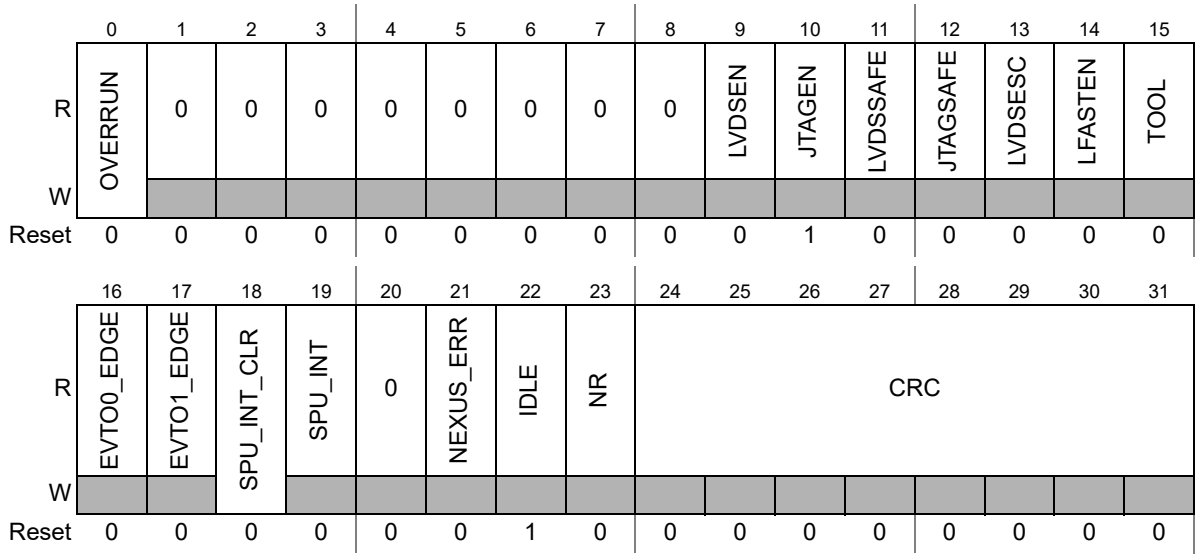


Figure 1070. JTAGM_SR register

The JTAGM_SR register is described in *Table 1024*.

Table 1024. JTAGM_SR register field descriptions

Field	Description
0 OVERRUN	Set by hardware when the JTAGM overwrites the first Tx data with the second Tx data. Writing 1 clears this bit. This bit is set only in LFAST mode.
1–8	Reserved
9 LVDSEN	DCI status: DCI LVDS port enable
10 JTAGEN	DCI status: DCI JTAG port enable
11 LVDSSAFE	DCI status: DCI LVDS safe mode
12 JTAGSAFE	DCI status: DCI JTAG safe mode
13 LVDSESC	DCI status: DCI LVDS escape mode
14	Reserved
15 TOOL	DCI status: DCI tool present, set when a debug tool is connected
16 EVTO0_EDGE	In software mode (DTM = 1), the bit stays set until cleared by writing to evto0_clr bit 0 No activity sensed on $\overline{\text{EVTO0}}$ 1 Activity defined by evto0_sense detected on $\overline{\text{EVTO0}}$

Table 1024. JTAGM_SR register field descriptions(Continued)

Field	Description
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.
20	Reserved
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 1 RDY has asserted indicating a Nexus RWA has completed bus access and is ready for the data register to be read **Note:** In LFAST mode (DTM = 0), this bit has no effect and should not be used.
24–31 CRC	Calculated CRC. CRC calculated on the information received. – Bit 24–31 reads as 0x00. Writing a 0 to any bit have no effect. – Writing 1 on bit 24 clears EVTO0_EDGE, bit 16 – Writing 1 on bit 25 clears EVTO1_EDGE, bit 17

62.4.1.3 Data out register 0 (JTAGM_DOR0)

Figure 1071 shows the format of the JTAGM_DOR0.

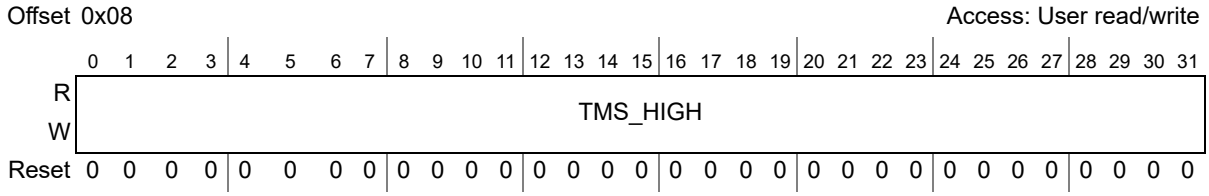


Figure 1071. JTAGM_DOR0 register

The JTAGM_DOR0 register is described in *Table 1025*.

Table 1025. 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

62.4.1.4 Data out register 1 (JTAGM_DOR1)

Figure 1072 shows the format of the JTAGM_DOR1.

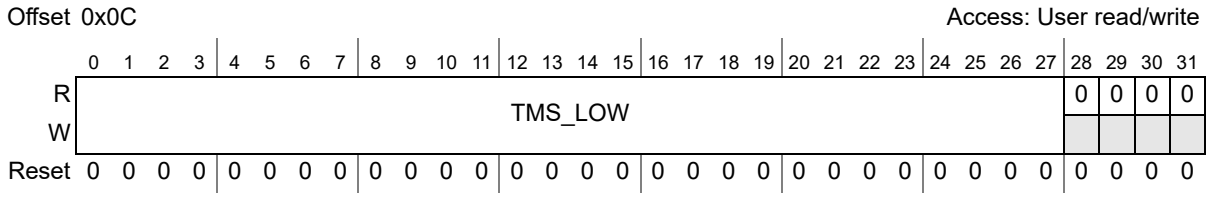


Figure 1072. JTAGM_DOR1 register

The JTAGM_DOR1 register is described in *Table 1026*.

Table 1026. 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.
28–31	Reserved

62.4.1.5 Data out register 2 (JTAGM_DOR2)

Figure 1073 shows the format of the JTAGM_DOR2.

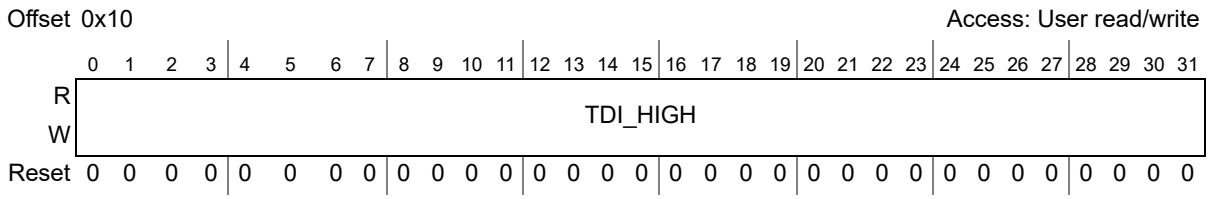


Figure 1073. JTAGM_DOR2 register

The JTAGM_DOR2 register is described in *Table 1027*.

Table 1027. 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

62.4.1.6 Data out register 3 (JTAGM_DOR3)

Figure 1074 shows the format of the JTAGM_DOR3.

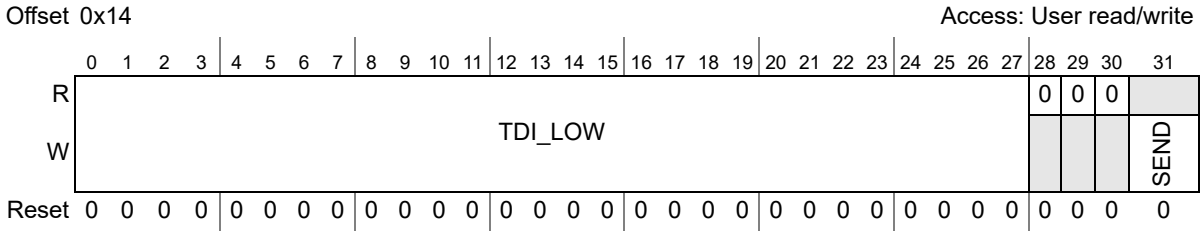


Figure 1074. JTAGM_DOR3 register

The JTAGM_DOR3 register is described in *Table 1028*.

Table 1028. 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
28–30	Reserved
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

62.4.1.7 Data input register 0 (JTAGM_DIR0)

Figure 1075 shows the format of the JTAGM_DIR0.

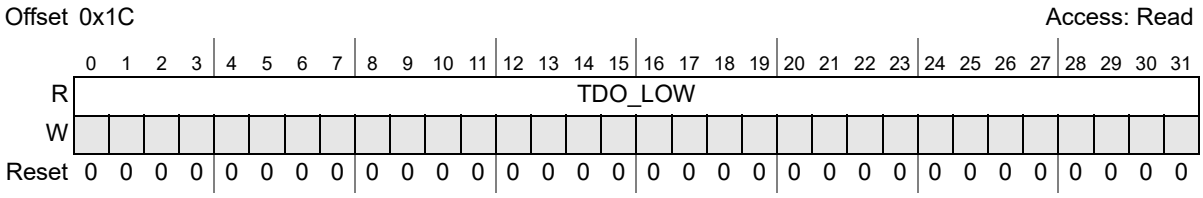


Figure 1075. JTAGM_DIR0 register

The JTAGM_DIR0 register is described in *Table 1029*.

Table 1029. JTAGM_DIR0 register field descriptions

Field	Description
0–31 TDO_LOW	Lower word of data received on TDO, bits 0–31

62.4.1.8 Data input register 1 (JTAGM_DIR1)

Figure 1076 shows the format of the JTAGM_DIR1.

Offset 0x20 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	TDO_HIGH																												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 1076. JTAGM_DIR1 register

The JTAGM_DIR1 register is described in *Table 1030*.

Table 1030. JTAGM_DIR1 register field descriptions

Field	Description
0–27 TDO_HIGH	Higher word of data received on TDO, bits 32–59
28–31	Reserved

63 Sequence Processing Unit (SPU)

63.1 Introduction

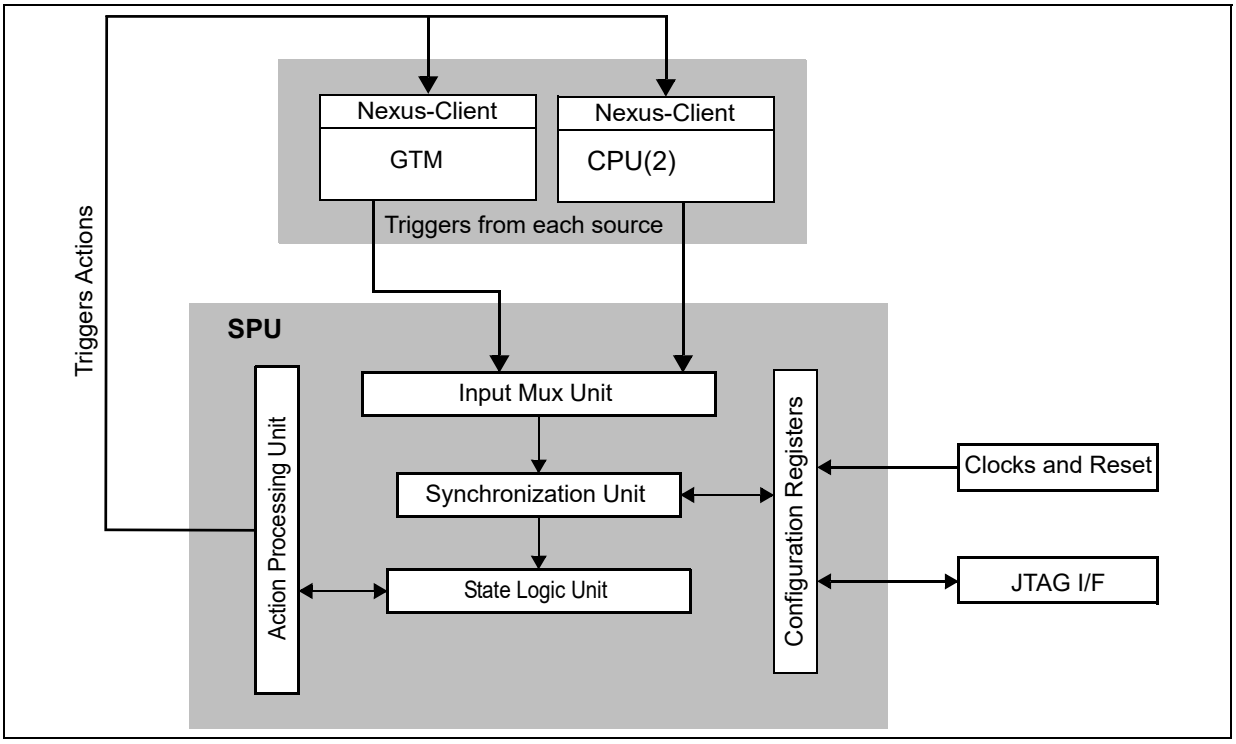
The Sequence Processing Unit (SPU) provides an on-device trigger functions similar to those found on a logic analyzer. In a complex SoC, performance monitor functions are available at two levels: system level and CPU level. Complex trigger and system performance monitor functions are implemented in the 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, GTM, 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.

63.1.1 SPU block diagram

Figure 1077 shows a top-level block diagram of the SPU.

Figure 1077. SPU block diagram



63.1.2 Overview

The SPU is able to generate complex debug events, based upon input events from sources (CPUs, GTM, 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 is 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 1112*.

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 1113*. 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 1114* and *Figure 1115*. 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 GTM and 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 1112*.
- 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 1113*.
- 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.

63.1.3 Features

The features of the SPU are as follows:

- clients:
  - 1 CPU
  - 1 GTM
- 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
  - Sixteen 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 GTM
  - SPU to/from NAR
  - SPU to/from DCI
  - SPU to/from JTAGM
- Interfaces
  - CPU
  - GTM
  - NAR
  - 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 63.2: SPU actions*

63.2 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 is as follows:

- Start/stop trace for a source
- Start/stop/increment counter/timer counter/timer[0–15]
- Reset counter/timer[0–15]
- Capture counter/timer Values[0–15]—place the specified counter/timer value into the trace stream for counters[0–15]
- Halt1—single signal (EVTI0) that can be used to halt a pre-configured DCI group1
- 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.

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.

63.3 Operation mode

63.3.1 SPU enable/disable

By default, the SPU is in disabled mode. In this state:

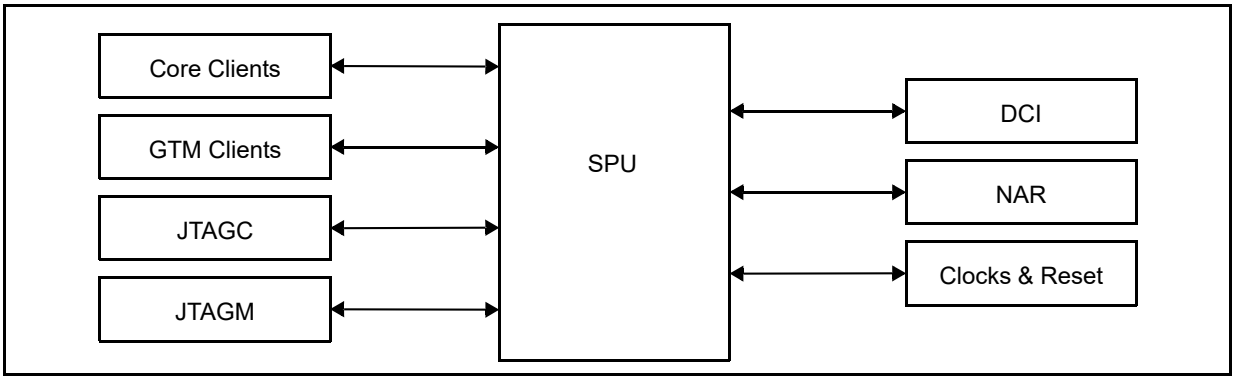
- All events/inputs are disabled such that no counting or other actions occur.
- All the SPU control registers are accessible.

The user must configure the control registers before enabling the SPU via the SPU Enable/Disable register.

63.4 SPU module interface

The SPU is connected to many clients such as the cores, GTM, system bus, NAR and Nexus clients. The SPU also interfaces to the DCI and the JTAG modules as shown in *Figure 1078*.

Figure 1078. SPU module interface



63.5 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 1031 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 1–0x49) registers is 4'h3
- Instruction to enable the Counter Control Unit (second set of registers with offsets 1–72) is 4'h2

Table 1031. SPU register summary

Offset (hex)	Register	Access	Reset Value	Location
1	Level1 Mux Selection 0 (L1SEL0)	R/W	32'h0	*on page 1787*
2	Level1 Mux Selection 1 (L1SEL1)	R/W	32'h0	*on page 1788*
3	Level1 Mux Selection 2 (L1SEL2)	R/W	32'h0	*on page 1788*
4	Level1 Mux Selection 3 (L1SEL3)	R/W	32'h0	*on page 1788*
5	Level1 Mux Selection 4(L1SEL4)	R/W	32'h0	*on page 1789*
6	Level1 Mux Selection 5 (L1SEL5)	R/W	32'h0	*on page 1789*
7	Level1 Mux Selection 6 (L1SEL6)	R/W	32'h0	*on page 1789*
8	Level1 Mux Selection 7 (L1SEL7)	R/W	32'h0	*on page 1790*
9	Reserved for future use	—	—	—
A	Reserved for future use	—	—	—
B	Input Trigger Level Detection 2 (ITLD2)	R/W	32'h0	*on page 1790*
C	Input Trigger Level Detection 3 (ITLD3)	R/W	32'h0	*on page 1791*
D	Reserved	—	—	—
E	Reserved	—	—	—
F	CPU2 Processor Except Vector Prefix (C2PEVP)	R/W	32'h0	*on page 1791*
10	Reserved for future use	—	—	—
11	Reserved for future use	—	—	—
12	Reserved	—	—	—
13	Reserved	—	—	—
14	CPU2 Interrupt Priority Selection (C2PIS)	R/W	32'h0	*on page 1792*
15	Reserved for future use	—	—	—
16	Reserved for future use	—	—	—
17	Level2 Mux State0 Selection 0 (L20SEL0)	R/W	32'h0	*on page 1793*
18	Level2 Mux State0 Selection 1 (L20SEL1)	R/W	32'h0	*on page 1794*
19	Level2 Mux State0 Selection 2 (L20SEL2)	R/W	32'h0	*on page 1796*
1A	Level2 Mux State0 Selection 3 (L20SEL3)	R/W	32'h0	*on page 1797*
1B	Level2 Mux State1 Selection 0 (L21SEL0)	R/W	32'h0	*on page 1793*
1C	Level2 Mux State1 Selection 1 (L21SEL1)	R/W	32'h0	*on page 1794*

Table 1031. SPU register summary(Continued)

Offset (hex)	Register	Access	Reset Value	Location
1D	Level2 Mux State1 Selection 2 (L21SEL2)	R/W	32'h0	on page 1796
1E	Level2 Mux State1 Selection 3 (L21SEL3)	R/W	32'h0	on page 1797
1F	Level2 Mux State2 Selection 0 (L22SEL0)	R/W	32'h0	on page 1793
20	Level2 Mux State2 Selection 1 (L22SEL1)	R/W	32'h0	on page 1794
21	Level2 Mux State2 Selection 2 (L22SEL2)	R/W	32'h0	on page 1796
22	Level2 Mux State2 Selection 3 (L22SEL3)	R/W	32'h0	on page 1797
23	Level2 Mux State3 Selection 0 (L23SEL0)	R/W	32'h0	on page 1793
24	Level2 Mux State3 Selection 1 (L23SEL1)	R/W	32'h0	on page 1794
25	Level2 Mux State3 Selection 2 (L23SEL2)	R/W	32'h0	on page 1796
26	Level2 Mux State3 Selection 3 (L23SEL3)	R/W	32'h0	on page 1797
27	Level2 Mux State4 Selection 0 (L24SEL0)	R/W	32'h0	on page 1793
28	Level2 Mux State4 Selection 1 (L24SEL1)	R/W	32'h0	on page 1794
29	Level2 Mux State4 Selection 2 (L24SEL2)	R/W	32'h0	on page 1796
2A	Level2 Mux State4 Selection 3 (L24SEL3)	R/W	32'h0	on page 1797
2B	Level2 Mux State5 Selection 0 (L25SEL0)	R/W	32'h0	on page 1793
2C	Level2 Mux State5 Selection 1 (L25SEL1)	R/W	32'h0	on page 1794
2D	Level2 Mux State5 Selection 2 (L25SEL2)	R/W	32'h0	on page 1796
2E	Level2 Mux State6 Selection 3 (L25SEL3)	R/W	32'h0	on page 1797
2F	Level2 Mux State6 Selection 0 (L26SEL0)	R/W	32'h0	on page 1793
30	Level2 Mux State6 Selection 1 (L26SEL1)	R/W	32'h0	on page 1794
31	Level2 Mux State6 Selection 2 (L26SEL2)	R/W	32'h0	on page 1796
32	Level2 Mux State6 Selection 3 (L26SEL3)	R/W	32'h0	on page 1797
33	Level2 Mux State7 Selection 0 (L27SEL0)	R/W	32'h0	on page 1793
34	Level2 Mux State7 Selection 1 (L27SEL1)	R/W	32'h0	on page 1794
35	Level2 Mux State7 Selection 2 (L27SEL2)	R/W	32'h0	on page 1796
36	Level2 Mux State7 Selection 3 (L27SEL3)	R/W	32'h0	on page 1797
37	Input/Output Inversion Control State0 (IOIC0)	R/W	32'h0	on page 1798
38	Input/Output Inversion Control State1 (IOIC1)	R/W	32'h0	on page 1798
39	Input/Output Inversion Control State2 (IOIC2)	R/W	32'h0	on page 1798
3A	Input/Output Inversion Control State3 (IOIC3)	R/W	32'h0	on page 1798
3B	Input/Output Inversion Control State4 (IOIC4)	R/W	32'h0	on page 1798
3C	Input/Output Inversion Control State5 (IOIC5)	R/W	32'h0	on page 1798
3D	Input/Output Inversion Control State6 (IOIC6)	R/W	32'h0	on page 1798
3E	Input/Output Inversion Control State7 (IOIC7)	R/W	32'h0	on page 1798

Table 1031. SPU register summary(Continued)

Offset (hex)	Register	Access	Reset Value	Location
3F	Sequence Control (SCTRL)	R/W	32'h0	on page 1800
40	State0 Goto Control (S0GC)	R/W	32'h0	on page 1802
41	State1 Goto Control (S1GC)	R/W	32'h0	on page 1802
42	State2 Goto Control (S2GC)	R/W	32'h0	on page 1802
43	State3 Goto Control (S3GC)	R/W	32'h0	on page 1802
44	State4 Goto Control (S4GC)	R/W	32'h0	on page 1802
45	State5 Goto Control (S5GC)	R/W	32'h0	on page 1802
46	State6 Goto Control (S6GC)	R/W	32'h0	on page 1802
47	State7 Goto Control (S7GC)	R/W	32'h0	on page 1802
48	SLU Status (SS)	R/W	32'h0	on page 1804
49	SPU Enable (SE)	R/W	32'h0	on page 1806
1	State0 True Action 0 (S0TA0)	R/W	32'h0	on page 1808
2	State0 True Action 1 (S0TA1)	R/W	32'h0	on page 1809
3	State1 True Action 0 (S1TA0)	R/W	32'h0	on page 1808
4	State1 True Action 1 (S1TA1)	R/W	32'h0	on page 1809
5	State2 True Action 0 (S2TA0)	R/W	32'h0	on page 1808
6	State2 True Action 1 (S2TA1)	R/W	32'h0	on page 1809
7	State3 True Action 0 (S3TA0)	R/W	32'h0	on page 1808
8	State3 True Action 1 (S3TA1)	R/W	32'h0	on page 1809
9	State4 True Action 0 (S4TA0)	R/W	32'h0	on page 1808
A	State4 True Action 1 (S4TA1)	R/W	32'h0	on page 1809
B	State5 True Action 0 (S5TA0)	R/W	32'h0	on page 1808
C	State5 True Action 1(S5TA1)	R/W	32'h0	on page 1809
D	State6 True Action 0 (S6TA0)	R/W	32'h0	on page 1808
E	State6 True Action 1(S6TA1)	R/W	32'h0	on page 1809
F	State7 True Action 0 (S7TA0)	R/W	32'h0	on page 1808
10	State7 True Action 1(S7TA1)	R/W	32'h0	on page 1809
11	State0 False Action 0 (S0FA0)	R/W	32'h0	on page 1810
12	State0 False Action 1 (S0FA1)	R/W	32'h0	on page 1810
13	State1 False Action 0 (S1FA0)	R/W	32'h0	on page 1810
14	State1 False Action 1(S1FA1)	R/W	32'h0	on page 1810
15	State2 False Action 0 (S2FA0)	R/W	32'h0	on page 1810
16	State2 False Action 1(S2FA1)	R/W	32'h0	on page 1810
17	State3 False Action 0 (S3FA0)	R/W	32'h0	on page 1810

Table 1031. SPU register summary(Continued)

Offset (hex)	Register	Access	Reset Value	Location
18	State3 False Action 1(S3FA0)	R/W	32'h0	*on page 1810*
19	State4 False Action 0 (S4FA0)	R/W	32'h0	*on page 1810*
1A	State4 False Action 1 (S4FA1)	R/W	32'h0	*on page 1810*
1B	State5 False Action 0 (S5FA0)	R/W	32'h0	*on page 1810*
1C	State5 False Action 1 (S5FA1)	R/W	32'h0	*on page 1810*
1D	State6 False Action 0 (S6FA0)	R/W	32'h0	*on page 1810*
1E	State6 False Action 1 (S6FA1)	R/W	32'h0	*on page 1810*
1F	State7 False Action 0 (S7FA0)	R/W	32'h0	*on page 1810*
20	State7 False Action 1 (S7FA1)	R/W	32'h0	*on page 1810*
21	DTM Trace Group Configuration (DTMTGC)	R/W	32'h0	*on page 1811*
22	PTM Trace Group Configuration (PTMTGC)	R/W	32'h0	*on page 1811*
23	OTM Trace Group Configuration (OTMTGC)	R/W	32'h0	*on page 1811*
24	WTM Trace Group Configuration (WTMTGC)	R/W	32'h0	*on page 1811*
25	Interrupt Status (INTS)	R/W	32'h0	*on page 1812*
26	Counter Control 0 (CCTRL0)	R/W	32'h0	*on page 1813*
27	Counter Control 1 (CCTRL1)	R/W	32'h0	*on page 1813*
28	Counter Control 2 (CCTRL2)	R/W	32'h0	*on page 1813*
29	Counter Control 3 (CCTRL3)	R/W	32'h0	*on page 1813*
2A	Counter Control 4 (CCTRL4)	R/W	32'h0	*on page 1813*
2B	Counter Control 5 (CCTRL5)	R/W	32'h0	*on page 1813*
2C	Counter Control 6 (CCTRL6)	R/W	32'h0	*on page 1813*
2D	Counter Control 7 (CCTRL7)	R/W	32'h0	*on page 1813*
2E	Counter Control 8 (CCTRL8)	R/W	32'h0	*on page 1813*
2F	Counter Control 9 (CCTRL9)	R/W	32'h0	*on page 1813*
30	Counter Control 10 (CCTRL10)	R/W	32'h0	*on page 1813*
31	Counter Control 11 (CCTRL11)	R/W	32'h0	*on page 1813*
32	Counter Control 12 (CCTRL12)	R/W	32'h0	*on page 1813*
33	Counter Control 13 (CCTRL13)	R/W	32'h0	*on page 1813*
34	Counter Control 14 (CCTRL14)	R/W	32'h0	*on page 1813*
35	Counter Control 15 (CCTRL15)	R/W	32'h0	*on page 1813*
36	Counter Compare 0 (CCMP0)	R/W	32'h0	*on page 1814*
37	Counter Compare 1 (CCMP1)	R/W	32'h0	*on page 1814*
38	Counter Compare 2 (CCMP2)	R/W	32'h0	*on page 1814*
39	Counter Compare 3 (CCMP3)	R/W	32'h0	*on page 1814*

Table 1031. SPU register summary(Continued)

Offset (hex)	Register	Access	Reset Value	Location
3A	Counter Compare 4 (CCMP4)	R/W	32'h0	*on page 1814*
3B	Counter Compare 5 (CCMP5)	R/W	32'h0	*on page 1814*
3C	Counter Compare 6 (CCMP6)	R/W	32'h0	*on page 1814*
3D	Counter Compare 7 (CCMP7)	R/W	32'h0	*on page 1814*
3E	Counter Compare 8 (CCMP8)	R/W	32'h0	*on page 1814*
3F	Counter Compare 9 (CCMP9)	R/W	32'h0	*on page 1814*
40	Counter Compare 10 (CCMP10)	R/W	32'h0	*on page 1814*
41	Counter Compare 11 (CCMP11)	R/W	32'h0	*on page 1814*
42	Counter Compare 12 (CCMP12)	R/W	32'h0	*on page 1814*
43	Counter Compare 13 (CCMP13)	R/W	32'h0	*on page 1814*
44	Counter Compare 14 (CCMP14)	R/W	32'h0	*on page 1814*
45	Counter Compare 15 (CCMP15)	R/W	32'h0	*on page 1814*
46	Counter Compare Status (CCOMS)	R/W	32'h0	*on page 1814*
47	Counter Overflow Status (COS)	R/W	32'h0	*on page 1815*
48	Counter Capture Status (CCAPS)	R/W	32'h0	*on page 1816*

63.5.1 Register descriptions

63.5.1.1 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 1112*.

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).

63.5.1.1.1 Level1 Mux Selection 0 (L1SEL0)

Figure 1079 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: User read/write

	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
R	0	0	0	0	0	0	0	0
W	MUX 7	MUX 6	MUX 5	MUX 4	MUX 3	MUX 2	MUX 1	MUX 0
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 1079. L1SEL0 register format

63.5.1.1.2 Level1 Mux Selection 1 (L1SEL1)

Figure 1080 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: User read/write

	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
R	0	0	0	0	0	0	0	0
W	MUX 15	MUX 14	MUX 13	MUX 12	MUX 11	MUX 10	MUX 9	MUX 8
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 1080. L1SEL1 register format

63.5.1.1.3 Level1 Mux Selection 2 (L1SEL2)

Figure 1081 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: User read/write

	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
R	0	0	0	0	0	0	0	0
W	MUX 23	MUX 22	MUX 21	MUX 20	MUX 19	MUX 18	MUX 17	MUX 16
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 1081. L1SEL2 register format

63.5.1.1.4 Level1 Mux Selection 3 (L1SEL3)

Figure 1082 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: User read/write

	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
R	0	0	0	0	0	0	0	0
W	MUX 31	MUX 30	MUX 29	MUX 28	MUX 27	MUX 26	MUX 25	MUX 24
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 1082. L1SEL3 register format

63.5.1.1.5 Level1 Mux Selection 4 (L1SEL4)

Figure 1083 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: User read/write

	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
R	0	0	0	0	0	0	0	0
W	MUX 39	MUX 38	MUX 37	MUX 36	MUX 35	MUX 34	MUX 33	MUX 32
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 1083. L1SEL4 register format

63.5.1.1.6 Level1 Mux Selection 5 (L1SEL5)

Figure 1084 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: User read/write

	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
R	0	0	0	0	0	0	0	0
W	MUX 47	MUX 46	MUX 45	MUX 44	MUX 43	MUX 42	MUX 41	MUX 40
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 1084. L1SEL5 register format

63.5.1.1.7 Level1 Mux Selection 6 (L1SEL6)

Figure 1085 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: User read/write

	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
R	0	MUX 55			0	MUX 54			0	MUX 53			0	MUX 52			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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1085. L1SEL6 register format

63.5.1.1.8 Level1 Mux Selection 7 (L1SEL7)

Figure 1086 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: User read/write

	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
R	0	MUX 63			0	MUX 62			0	MUX 61			0	MUX 60			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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1086. L1SEL7 register format

63.5.1.1.9 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 1087* shows the format of the ITLD2 register. All of the MUX fields in the ITLD2 register have the same description as shown in *Table 1032*.

Offset 0x0B Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Mux47		Mux46		Mux45		Mux44		Mux43		Mux42		Mux41		Mux40	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Mux39		Mux38		Mux37		Mux36		Mux35		Mux34		Mux33		Mux32	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1087. ITLD2 register format

The ITLD registers Mux field is described in *Table 1032*.

Table 1032. ITLD register field descriptions

Field	Description
Mux*	Mux Output Signal Detection. 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

63.5.1.1.10 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 1088* shows the format of the ITLD3 register. All of the MUX fields in the ITLD3 register have the same description as shown in *Table 1032*.

Offset 0x0C — Access: User read/write

	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
R W	Mux63	Mux62	Mux61	Mux 60	Mux59	Mux58	Mux57	Mux56
Reset	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0

	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
R W	Mux55	Mux54	Mux53	Mux52	Mux51	Mux50	Mux49	Mux48
Reset	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0

Figure 1088. ITLD3 register format

63.5.1.1.11 CPU2 processor exception vector prefix (C2PEVP)

The SPU captures the exception vector from CPU2 when the exception enable signal is asserted. The exception vector decoding is defined in *Table 1056*. The SPU compares this captured vector with the value in the C2PEVP register. *Figure 1089* shows the format of the C2PEVP register.

Offset 0x0F — Access: User read/write

	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
W	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	CPU2			
R								
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 1089. C2PEVP register format

The C2PEVP register is described in *Table 1033*.

Table 1033. C2PEVP register field descriptions

Field	Description
15–0 CPU2	CPU2 processor exception vector. Program this field with one of the vector values from *Table 1056* to select the corresponding processor exception. **Note:** This field must use the byte swapped values in the CnPEVP register CPUn field value column in *Table 1056*.
31–16	Reserved

63.5.1.1.12 CPU2 interrupt priority selection (C2PIS)

Figure 1090 shows the format of the C2PIS register. The SPU compares the input CPR vector from CPU2 with the value programmed to the C2PIS 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 0x14 Access: User read/write

	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
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU2							
R																																
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 1090. C2PIS register format

The C2PIS register is described in *Table 1034*.

Table 1034. C2PIS register field descriptions

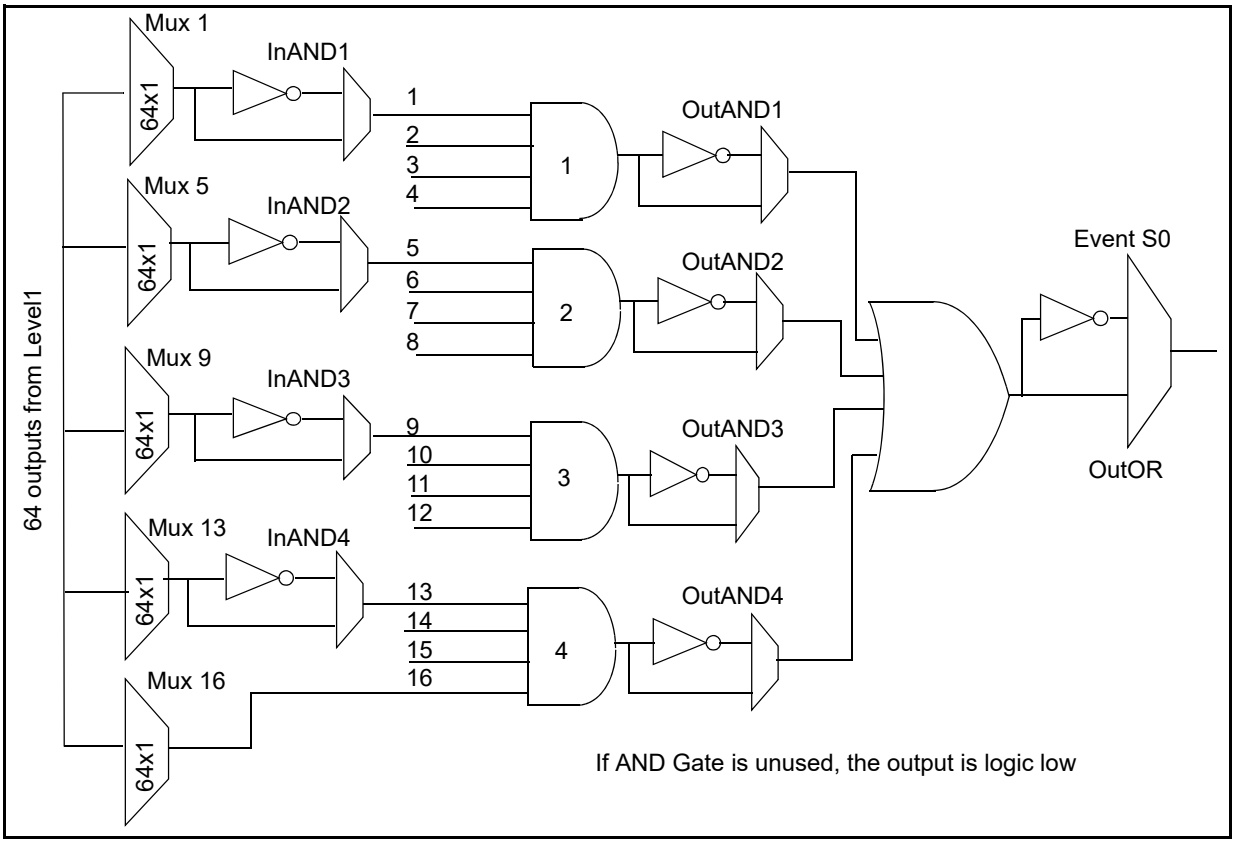
Field	Description
7–0 CPU2	CPU2 Priority Interrupt Selection.
31–8	Reserved

63.5.1.2 Level2 input Mux configurations registers

At level1, the set of input Mux control registers (described in *Section 63.5.1.1: 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 1112* for State0. Similar logic is provided for States 1–7.

Figure 1091. 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.

63.5.1.2.1 Level2 Mux state n selection 0 (L2nSEL0)

Figure 1092 shows the format of the L2nSEL0 register where the state number, n = 0–7.

Offset: 0x17 (State0), 0x1B (State1), 0x1F (State2), 0x23 (State3), 0x27 (State4), 0x2B (State5), 0x2F (State6), 0x33 (State7)

Access: User read/write

	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
R	0	0	FirstANDInput4						0	0	FirstANDInput3						0	0	FirstANDInput2						0	0	FirstANDInput1					
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 1092. L2nSEL0 register format

The L2nSEL0 register fields are described in *Table 1035*.

Table 1035. L2nSEL0 register field descriptions

Field	Description
5–0 FirstANDInput1	First AND Gate Input1 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
7–6	Reserved. Read returns 0.
13–8 FirstANDInput2	First AND Gate Input 2 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
15–14	Reserved. Read returns 0.
21–16 ANDInput3	First AND Gate Input 3 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
23–22	Reserved. Read returns 0.
29–24 FirstANDInput4	First AND Gate Input 4 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
31–30	Reserved. Read returns 0.

63.5.1.2.2 Level2 Mux state n selection 1 (L2nSEL1)

Figure 1093 shows the format of the L2nSEL1 register where the state number, n = 0–7.

Offset: 0x18 (State0), 0x1C (State1), 0x20 (State2), 0x24 (State3), 0x28 (State4), 0x2C (State5), 0x30 (State6), 0x34 (State7)

Access: User read/write

	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
R	0	0	SecondANDInput4						0	0	SecondANDInput3						0	0	SecondANDInput2						0	0	SecondANDInput1					
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 1093. L2nSEL1 register format

The L2nSEL1 register fields are described in *Table 1036*.

Table 1036. L2nSEL1 register field descriptions

Field	Description
5–0 SecondANDInput1	Second AND Gate Input1 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
7–6	Reserved. Read returns 0.
13–8 SecondANDInput2	Second AND Gate Input 2 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
15–14	Reserved. Read returns 0.
21–16 SecondANDInput3	Second AND Gate Input 3 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
23–22	Reserved. Read returns 0.

Table 1036. L2nSEL1 register field descriptions(Continued)

Field	Description
29–24 SecondANDInput4	Second AND Gate Input 4 Selection. 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
31–30	Reserved. Read returns 0.

63.5.1.2.3 Level2 Mux state n selection 2 (L2nSEL2)

Figure 1094 shows the format of the L2nSEL2 register where the state number, n = 0–7.

Offset: 0x19 (State0), 0x1D (State1), 0x21 (State2), 0x25 (State3), 0x29 (State4), 0x2D (State5), 0x31 (State6), 0x35 (State7)

Access: User read/write

	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
R	0	0	ThirdANDInput4						0	0	ThirdANDInput3						0	0	ThirdANDInput2						0	0	ThirdANDInput1					
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 1094. L2nSEL2 register format

The L2nSEL2 register fields are described in *Table 1037*.

Table 1037. L2nSEL2 register field descriptions

Field	Description
5–0 ThirdANDInput1	Third AND Gate Input1 Selection. 000000No Input is selected (tie to 1’b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
7–6	Reserved. Read returns 0.

Table 1037. L2nSEL2 register field descriptions(Continued)

Field	Description
13–8 ThirdANDInput2	Third AND Gate Input 2 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
15–14	Reserved. Read returns 0.
21–16 ThirdANDInput3	Third AND Gate Input 3 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
23–22	Reserved. Read returns 0.
29–24 ThirdANDInput4	Third AND Gate Input 4 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
31–30	Reserved. Read returns 0.

63.5.1.2.4 Level2 Mux state n selection 3 (L2nSEL3)

Figure 1095 shows the format of the L2nSEL3 register where the state number, n = 0–7.

Offset 0x1A (State0), 0x1E (State1), 0x22 (State2), 0x26 (State3), 0x2A (State4), 0x2E (State5), 0x32 (State6), 0x36 (State7)

Access: User read/write

	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
R	0	0	FourthANDInput4						0	0	FourthANDInput3						0	0	FourthANDInput2						0	0	FourthANDInput1					
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 1095. L2nSEL3 register format

The L2nSEL3 register fields are described in *Table 1038*.

Table 1038. L2nSEL3 register field descriptions

Field	Description
5–0 FourthANDInput1	Fourth AND Gate Input1 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
7–6	Reserved. Read returns 0.
13–8 FourthANDInput2	Fourth AND Gate Input 2 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
15–14	Reserved. Read returns 0.
21–16 FourthANDInput3	Fourth AND Gate Input 3 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
23–22	Reserved. Read returns 0.
29–24 FourthANDInput4	Fourth AND Gate Input 4 Selection. 000000No Input is selected (tie to 1'b1) 000001Level1 MUX 0 output 000010Level1 MUX 1 output 000011Level1 MUX 2 output … 111110Level1 MUX 61 output 111111Level1 MUX 62 output
31–30	Reserved. Read returns 0.

63.5.1.3 Sequence unit registers

63.5.1.3.1 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 1096* shows the format of the IOICn register where the state number, n = 0–7.

Offset: 0x37 (State0), 0x38 (State1), 0x39 (State2), 0x3A (State3), 0x3B (State4), 0x3C (State5), 0x3D (State6), 0x3E (State7)

Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	OutOR	OutAND4	OutAND3	OutAND2	OutAND1	InAND 4	InAND 3	InAND 2	InAND1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1096. IOICn register format

The IOICn register fields are described in *Table 1039*.

Table 1039. IOICn register field descriptions

Field	Description
0 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
1 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
2 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
3 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
4 OutAND1	Output AND Gate 1 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
5 OutAND2	Output AND Gate 2 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event

Table 1039. IOICn register field descriptions(Continued)

Field	Description
6 OutAND3	Output AND Gate 3 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
7 OutAND4	Output of AND Gate 4 Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
8 OutOR	Output OR GATE Control. 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
9–31	Reserved. Read returns 0.

63.5.1.3.2 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 1097* shows the format of the SCTRL register.

Offset 0x3F Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	ST2			ST1			ST0	0	ST2			ST1			ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	ST2			ST1			ST0	0	ST2			ST1			ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1097. SCTRL register format

The SCTRL register fields are described in *Table 1040*.

Table 1040. SCTRL register field descriptions

Field	Description
0 ST0	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
3–1 ST1	Sequence 1 Start State. 000 State 0 (Always)

Table 1040. SCTRL register field descriptions(Continued)

Field	Description
6–4 ST2	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
7	Reserved. Read returns 0.
8 ST0	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
11–9 ST1	Sequence 2 Start State.(1) 000State 0 is the Start state for sequence 2 001State 1 is the Start state for sequence 2 010State 2 is the Start state for sequence 2 011State 3 is the Start state for sequence 2 100State 4 is the Start state for sequence 2 101State 5 is the Start state for sequence 2 110State 6 is the Start state for sequence 2 111State 7 is the Start state for sequence 2
14–12 ST2	Sequence 2 End State. 000State 0 is the Stop state for sequence 2 001State 1 is the Stop state for sequence 2 010State 2 is the Stop state for sequence 2 011State 3 is the Stop state for sequence 2 100State 4 is the Stop state for sequence 2 101State 5 is the Stop state for sequence 2 110State 6 is the Stop state for sequence 2 111State 7 is the Stop state for sequence 2
15	Reserved. Read returns 0.
16 ST0	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
19–17 ST1	Sequence 3 Start State.(1) 000State 0 is the Start state for sequence 3 001State 1 is the Start state for sequence 3 010State 2 is the Start state for sequence 3 011State 3 is the Start state for sequence 3 100State 4 is the Start state for sequence 3 101State 5 is the Start state for sequence 3 110State 6 is the Start state for sequence 3 111State 7 is the Start state for sequence 3

Table 1040. SCTRL register field descriptions(Continued)

Field	Description
22–20 ST2	Sequence 3 End State. 000State 0 is the Stop state for sequence 3 001State 1 is the Stop state for sequence 3 010State 2 is the Stop state for sequence 3 011State 3 is the Stop state for sequence 3 100State 4 is the Stop state for sequence 3 101State 5 is the Stop state for sequence 3 110State 6 is the Stop state for sequence 3 111State 7 is the Stop state for sequence 3
23	Reserved. Read returns 0.
24 ST0	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
27–25 ST1	Sequence 4 Start State.(1) 000State 0 is the Start state for sequence 4 001State 1 is the Start state for sequence 4 010State 2 is the Start state for sequence 4 011State 3 is the Start state for sequence 4 100State 4 is the Start state for sequence 4 101State 5 is the Start state for sequence 4 110State 6 is the Start state for sequence 4 111State 7 is the Start state for sequence 4
30–28 ST2	Sequence 4 End State. 000State 0 is the Stop state for sequence 4 001State 1 is the Stop state for sequence 4 010State 2 is the Stop state for sequence 4 011State 3 is the Stop state for sequence 4 100State 4 is the Stop state for sequence 4 101State 5 is the Stop state for sequence 4 110State 6 is the Stop state for sequence 4 111State 7 is the Stop state for sequence 4
31	Reserved. Read returns 0.

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.

63.5.1.3.3 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 1098 shows the format of the SnGC register where the state number, n = 0–7.

0x40 (State0)
0x41 (State1)
0x42 (State2)
Offset 0x43 (State3)
0x44 (State4)
0x45 (State5)
0x46 (State6)
0x47 (State7)

Access: User read/write

	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
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	NF			FE	NT			TE
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 1098. SnGC register format

The SnGC register fields are described in *Table 1041*.

Table 1041. SnGC register field descriptions

Field	Description
0 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
3–1 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
4 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 1041. SnGC register field descriptions(Continued)

Field	Description
7–5 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
31–8	Reserved. Read returns 0.

63.5.1.4 SLU status (SS) register

The SLU status register indicates the states that have been hit/encountered in any of the active sequences. *Figure 1099* shows the format of the SS register.

Offset 0x48 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	IM0	EM2	0	0	SAS4			SAS3
W									w1c	w1c			w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAS3		SAS2			SAS1			SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
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 1099. SS register format

The SS register fields are described in *Table 1042*.

Table 1042. SS register field descriptions

Field	Description
0 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
1 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
2 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

Table 1042. SS register field descriptions(Continued)

Field	Description
3 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
4 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
5 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
6 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
7 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
10 –8 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
13 –11 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
16–14 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

Table 1042. SS register field descriptions(Continued)

Field	Description
19 –17 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
20	Reserved
21	Reserved
22 EM2	Exception Match 2. This bit indicates the status of the processor exception vector match value of CPU2. 0 No match 1 Match occurred
23	Reserved
24	Reserved
25 IM2	Interrupt Match 2. This bit indicates the status of the match value on interrupt priority register of CPU2. 0 No match 1 Match occurred
31–26	Reserved. Read returns 0

63.5.1.5 SPU enable (SE) register

The SPU enable/disable register is used to enable and disable the SPU. If disabled, all inputs are masked. *Figure 1100* shows the format of the SE register.

Offset 0x49 Access: User read/write

	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
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	0	0	0	SED
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 1100. SE register format

The SE register fields are described in *Table 1043*.

Table 1043. SE register field descriptions

Field	Description
0 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

Table 1043. SE register field descriptions(Continued)

Field	Description
1	Reserved
31–2	Reserved

63.5.1.6 SLU action unit registers

The action unit of the SPU is responsible for generating the action triggers supported by various actions, listed in *Section 63.2: SPU actions*. 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 1044* shows the bit mapping for all the possible actions divided in three groups: client based, counter based, system related.

Table 1044. 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	Reserved	0000	No action	—	0=Stop 1=Start
		00001	Reserved	—	—	Valid for Cores, GTM and trace group actions.	
		00010	CPU2	0010	OTM/gtm_gen1 trace		
		00011–00111	Reserved	0011	PTM/gtm_gen2 trace		
		01000	GTM	0100	WTM/gtm_gen3 trace		
		01001–01111	Reserved	0101	NAR_supp_trig1	Valid for NAR as the suppress trigger actions	For NAR suppress trigger = Don't care
		10000	Reserved	0110	NAR_supp_trig2		
		10001	Reserved	0111	NAR_supp_trig3		
		10010–10111	Reserved	1000	NAR_supp_trig4		
		11000	NAR	1001	GTM_dtm_tr0	Valid for GTM	0=Stop 1=Start
		11001–11110	Reserved	1010	GTM_dtm_tr1		
				1011	GTM_ftm_tr0		
				1100	GTM_ftm_tr1		
		11111	Trace group(1)	1101–1111	Reserved	Valid for CPU cores	
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 1044. 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	NAR_sync_trigg	Valid for NAR	
				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.

63.5.1.6.1 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 1044*. 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 1101 shows the format of the SnTA0 register where the state number, n = 0–7.

Offset
0x01 (State0)
0x03 (State1)
0x05 (State2)
0x07 (State3)
0x09 (State4)
0x0B (State5)
0x0D (State6)
0x0F (State7)

Access: User read/write

	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
R	0	0	0	0	ACTION2												0	0	0	0	ACTION1											
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 1101. SnTA0 register format

The SnTA0 register fields are described in *Table 1045*.

Table 1045. SnTA0 register field descriptions

Field	Description
0–11 ACTION1	Action 1 to be performed for State n true condition. The decoding of these bits is defined in *Table 1044*
12–15	Reserved
16–27 ACTION2	Action 2 to be performed for State n true condition. The decoding of these bits is defined in *Table 1044*
28–31	Reserved

63.5.1.6.2 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 1044*. 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 1102* shows the format of the SnTA1 register where the state number, n = 0–7.

Offset: 0x02 (State0), 0x04 (State1), 0x06 (State2), 0x08 (State3), 0x0A (State4), 0x0C(State5), 0x0E (State6), 0x10 (State7)

Access: User read/write

	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
R	0	0	0	0	ACTION4												0	0	0	0	ACTION3											
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 1102. SnTA1 register format

The SnTA1 register fields are described in *Table 1046*.

Table 1046. SnTA1 register field descriptions

Field	Description
0–11 ACTION3	Action 3 to be performed for State n true condition. The decoding of these bits is defined in *Table 1044*
12–15	Reserved
16–27 ACTION4	Action 4 to be performed for State n true condition. The decoding of these bits is defined in *Table 1044*
28–31	Reserved

63.5.1.6.3 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 1044*. 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 1103* shows the format of the SnFA0 register where the state number, n = 0–7.

Offset: 0x11 (State0), 0x13 (State1), 0x15 (State2), 0x17 (State3), 0x19 (State4), 0x1B (State5), 0x1D (State6), 0x1F (State7)

Access: User read/write

	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
R	0	0	0	0	ACTION2												0	0	0	0	ACTION1											
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 1103. SnFA0 register format

The SnFA0 register fields are described in *Table 1047*.

Table 1047. SnFA0 register field descriptions

Field	Description
0–11 ACTION1	Action 1 to be performed for State n false condition. The decoding of these bits is defined in *Table 1044*
12–15	Reserved
16–27 ACTION2	Action 2 to be performed for State n false condition. The decoding of these bits is defined in *Table 1044*
28–31	Reserved

63.5.1.6.4 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 1044*. 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 1104* shows the format of the SnFA1 register where the state number, n = 0–7.

Offset: 0x12 (State0), 0x14 (State1), 0x16 (State2), 0x18 (State3), 0x1A (State4), 0x1C(State5), 0x1E (State6), 0x20 (State7)

Access: User read/write

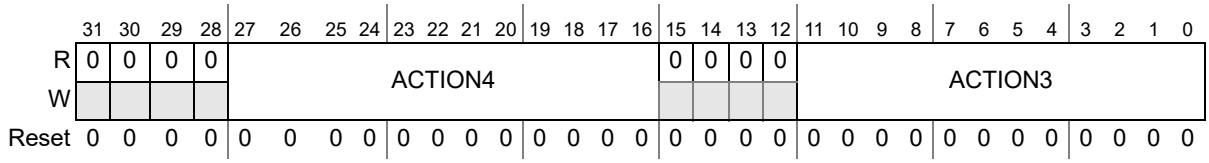


Figure 1104. SnFA1 register format

The SnFA1 register fields are described in *Table 1048*.

Table 1048. SnFA1 register field descriptions

Field	Description
0–11 ACTION3	Action 3 to be performed for State n false condition. The decoding of these bits is defined in *Table 1044*
12–15	Reserved
16–27 ACTION4	Action 4 to be performed for State n false condition. The decoding of these bits is defined in *Table 1044*
28–31	Reserved

63.5.1.6.5 Trace group configuration registers (DTMGC, PTMGC, OTMGC and WTMGC)

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 (DTMGC) register
- PTM trace—clients are configured with the PTM trace group configuration (PTMGC) register
- OTM trace—clients are configured with the OTM trace group configuration (OTMGC) register
- WTM trace—clients are configured with the WTM trace group configuration (WTMGC) register

Each of these groups is specific to a trace. Any number of clients can be enabled for one trace using the action trace group.

Figure 1105 shows the format of the trace group configuration registers.

Offset 0x21 (DTMGC)
0x22 (PTMGC)
0x23 (OTMGC)
0x24 (WTMGC)

Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU2	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1105. Trace group configuration register format

The trace group configuration register fields are described in *Table 1049*.

Table 1049. Trace group configuration register field descriptions

Field	Description
0	Reserved
1	Reserved
2 CPU2	0 Disable trace for CPU2 1 Enable trace for CPU2
3–15	Reserved
16–31	Reserved

63.5.1.6.6 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 1106* shows the format of the INTS register.

Offset 0x25

Access: User read/write

	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
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	0	0	0	INT
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 1106. INTS register format

The INTS register fields are described in *Table 1050*.

Table 1050. INTS register field descriptions

Field	Description
0 INT	1 Interrupt status bit is cleared from JTAGM
31–1	Reserved

63.5.1.7 Counters control n (CCTRLn) registers

There are 16 configurable counters/timers in the SPU. Configuration control of these counters is defined in the counter control registers. *Figure 1107* shows the format of the CCTRLn registers where the counter number, n = 0–15.

Offset 0x26–0x35 (CCTRL0–15) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 1107. CCTRLn register format

The CCTRLn register fields are described in *Table 1051*.

Table 1051. CCTRLn register field descriptions

Field	Description
0 ENABLE	Enable. 0 Counter disabled 1 Counter enabled
1 MODE	Mode Select. Select counter or timer operation. 0 Counter 1 Timer
2 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.
3	Reserved. Read returns 0.

Table 1051. CCTRLn register field descriptions(Continued)

Field	Description
6–4 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
7 SW_RESET	Reset. Counter can be reset by software 0 Software reset is disabled 1 Software reset
8 Reserved	This bit is for internal use only and must be set to 0.
31–9	Reserved. Read returns 0.

63.5.1.8 Counters compare n (CCMPn) registers

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 1108* shows the format of the CCMPn registers where the counter number, n = 0–15.

Offset 0x36–0x45 (CCMP0–15) Access: User read/write

	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
R W	COMPARE_VAL																															
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 1108. CCMPn register format

The CCMPn register fields are described in *Table 1052*.

Table 1052. CCMPn register field descriptions

Field	Description
31–0 COMPARE_VAL	Compare value. This value is compared with the corresponding counter/timer value to generate an event for the SPU.

63.5.1.9 Status registers

63.5.1.9.1 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 1109* shows the format of the CCOMS register. All of the CC fields in the CCOMS register have the same description as shown in *Table 1053*.

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: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 1109. CCOMS register format

The CCOMS register fields are described in *Table 1053*, where n is the counter number.

Table 1053. CCOMS register field descriptions

Field	Description
CCn	Counter n Compare. 0 Counter status is not true 1 Counter status is true

63.5.1.9.2 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 1110* shows the format of the COS register. All of the CO fields in the COS register have the same description as shown in *Table 1054*.

Offset 0x47 Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 1110. COS register format

The COS register fields are described in *Table 1054*, where n is the counter number.

Table 1054. COS register field descriptions

Field	Description
COn	Counter n Overflow. 0 Counter did not overflow. 1 Counter overflow occurred.

63.5.1.9.3 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 1111* shows the format of the CCAPS register.

All of the CC fields in the CCAPS register have the same description as shown in *Table 1055*.

Offset 0x48 Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 1111. CCAPS register format

The CCAPS register fields are described in *Table 1055*, where n is the counter number.

Table 1055. CCAPS register field descriptions

Field	Description
CCn	Counter n Captured. 0 Counter not captured. 1 Counter captured.

63.6 Functional description

63.6.1 Input Mux unit

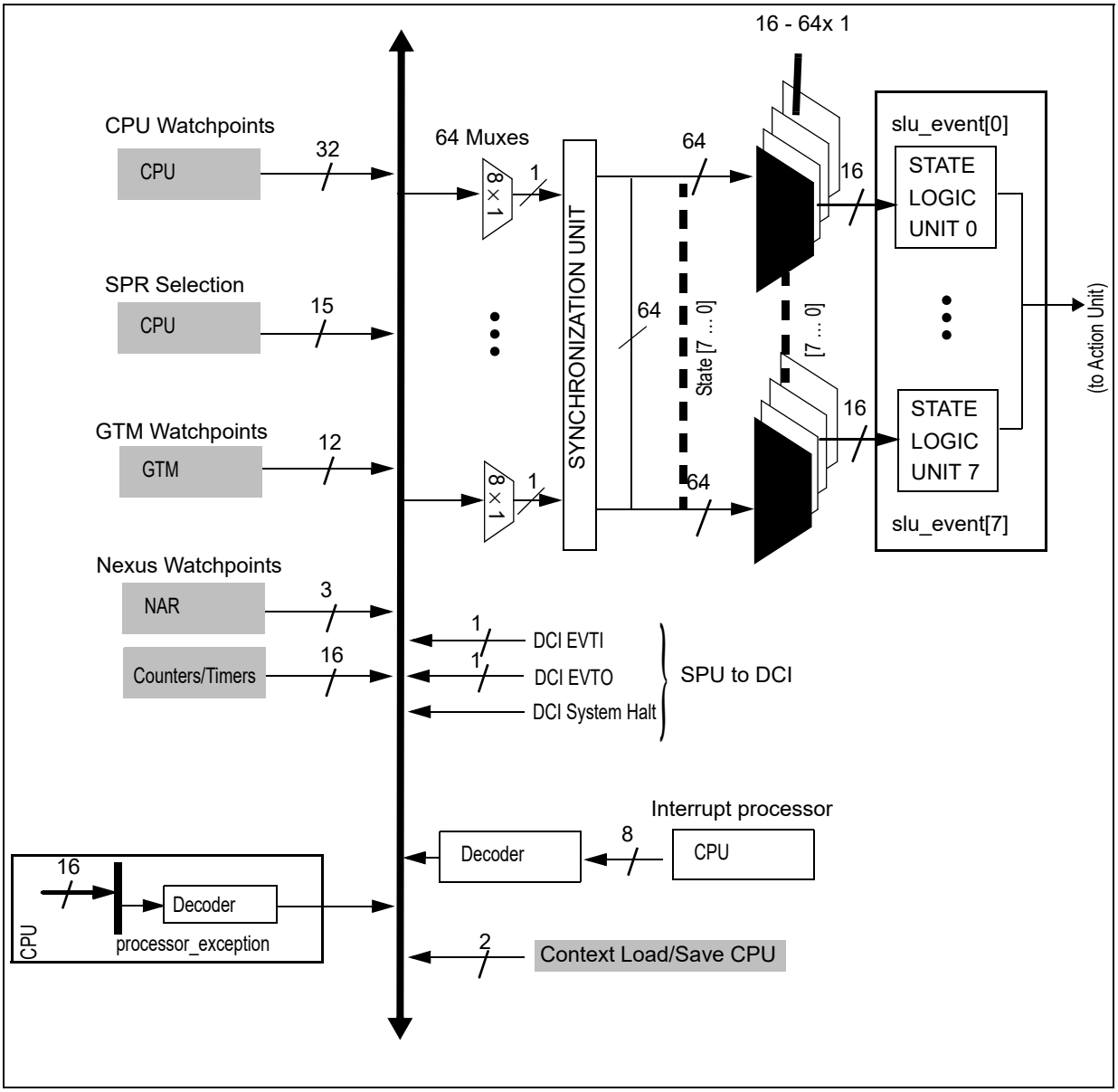
The SPU receives numerous inputs in the form of complex triggers from various clients to create a predefined sequence. One state logic can accommodate 16 inputs at one point of time. Therefore, an input Muxing unit becomes essential for reducing the debug and performance events to a manageable set of 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 1112*.

- 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 is passed to the second level of muxing. The selection mechanism is shown in *Figure 1112*.
- 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 1112*.

Figure 1112. Input Mux unit block diagram



63.6.2 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 1056*. The CPU exception vector values from the CPUs must be byte swapped before they are programmed to the CPUn field of the CnPEVP (n = 0–0) registers as shown in *Table 1056*.

Table 1056. Processor exception processing conditions

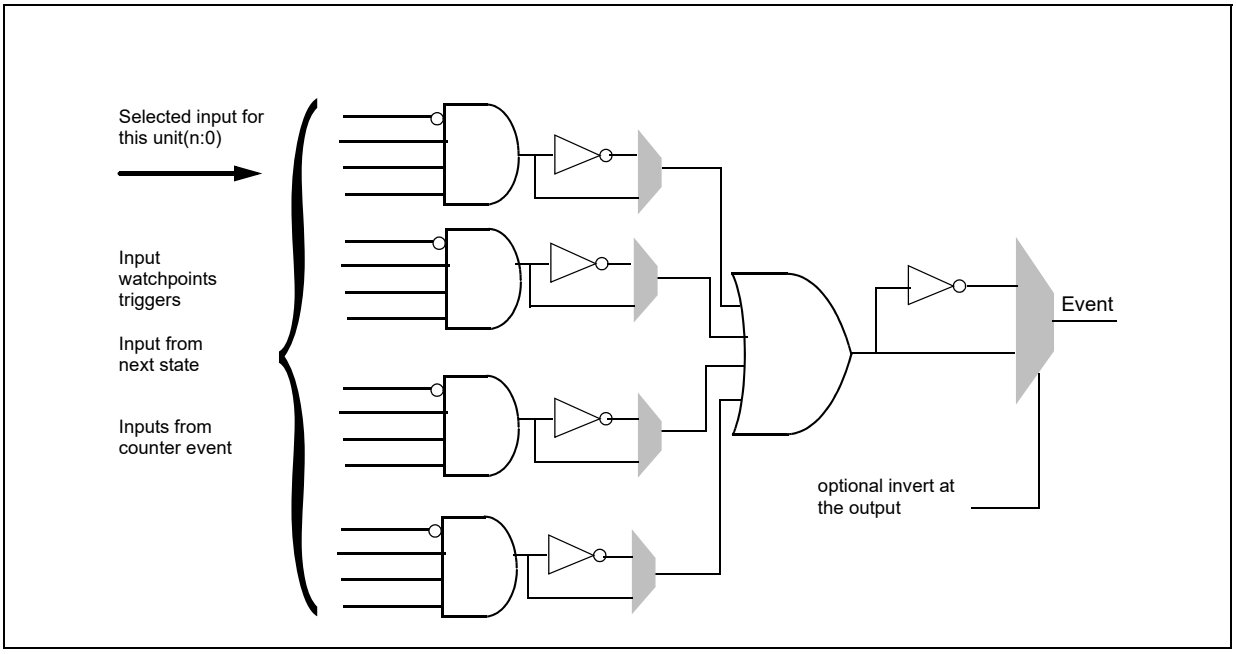
CPU exception vector	CnPEVP register CPUn field value	Exception condition
0x0000	0x0000	Critical Input—autovectored
0x0010	0x1000	Machine check
0x0020	0x2000	Data storage
0x0030	0x3000	Instruction storage
0x0040	0x4000	External input—autovectored
0x0050	0x5000	Alignment
0x0060	0x6000	Program trap
0x0061	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

63.6.3 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 1113*.

Figure 1113. 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.

63.6.4 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 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.

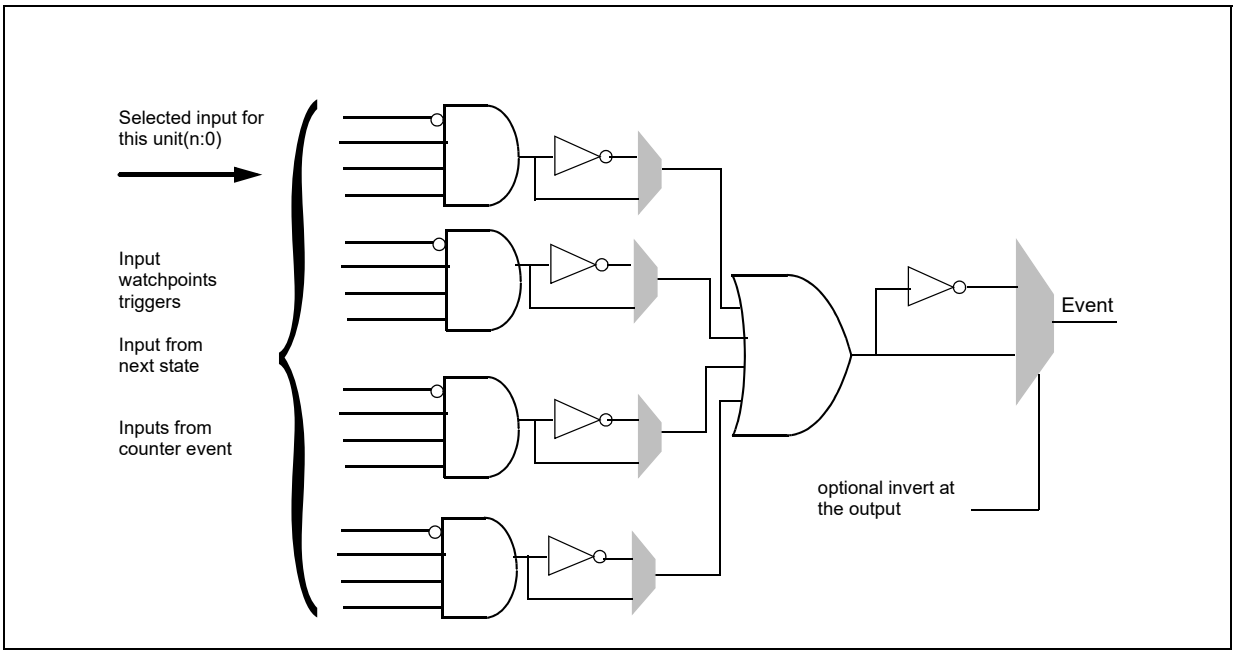
For example, the following state configurations are possible:

- 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 1114*:

- 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

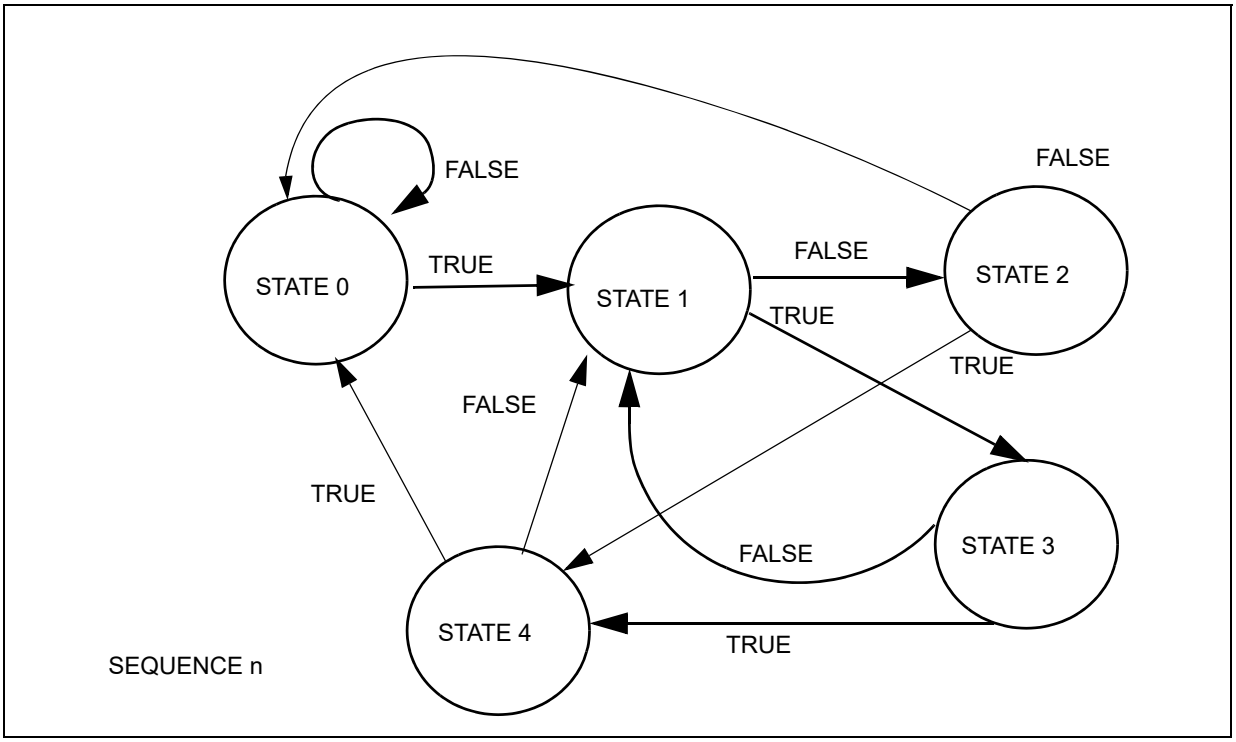
Figure 1114. Example of a simple configurable if-then-else sequence



63.6.4.1 Complex sequence example

A user can create a complex sequence using five states as shown in the state transition diagram in *Figure 1115*.

Figure 1115. Complex sequence example



63.6.4.2 Typical sequence example

For this example, the user would like to create a timestamp in the trace stream each time a function is entered and stop the trace signal on the 12th occurrence of the function. This use case can be implemented as follows:

- State 0 takes an instruction address compare as its input. On a match of the instruction address to a specific function, called function_x, the sequence moves to State 1. Upon this action, State0 is configured to insert a timestamp into the trace stream and also increment an SPU counter, say counter[0].
- The sequence is now in State 1. State 1 takes an input from counter[0]. Counter[0] has been pre-configured to create a match event when the counter value is equal to 12.
  - If the value of counter[0] is less than 12, then the sequence moves back to State0.
  - If the value of couner[0] is equal to 12, then the sequence completes and an action from State1 disables the trace signal.

63.6.4.3 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:

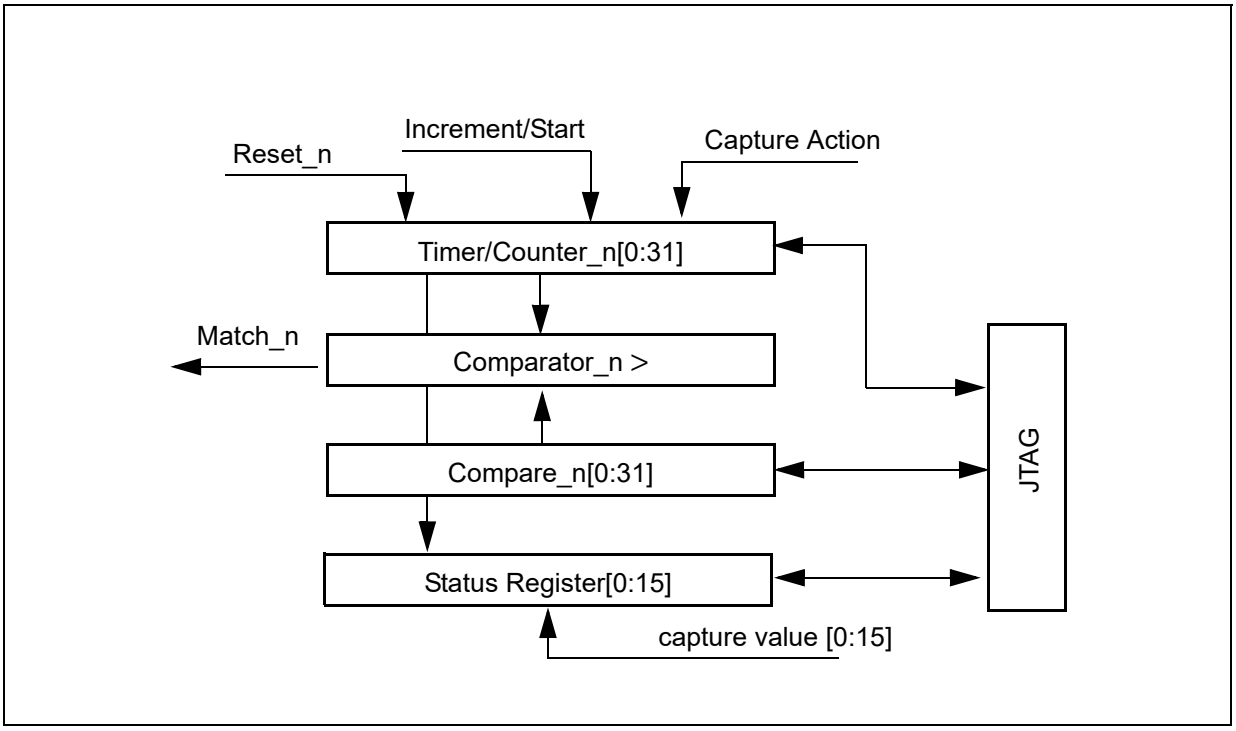
- States—each state has a sticky bit that is set upon a true match of the state logic. This sticky bit is cleared upon a write to this register.
- Sequence status—each sequence has a field indicating the active state of the state machine. This is a read only field.

63.6.5 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 1116*. 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 1116. Counters/timers overview



63.6.5.1 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 if 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.

- Events—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.

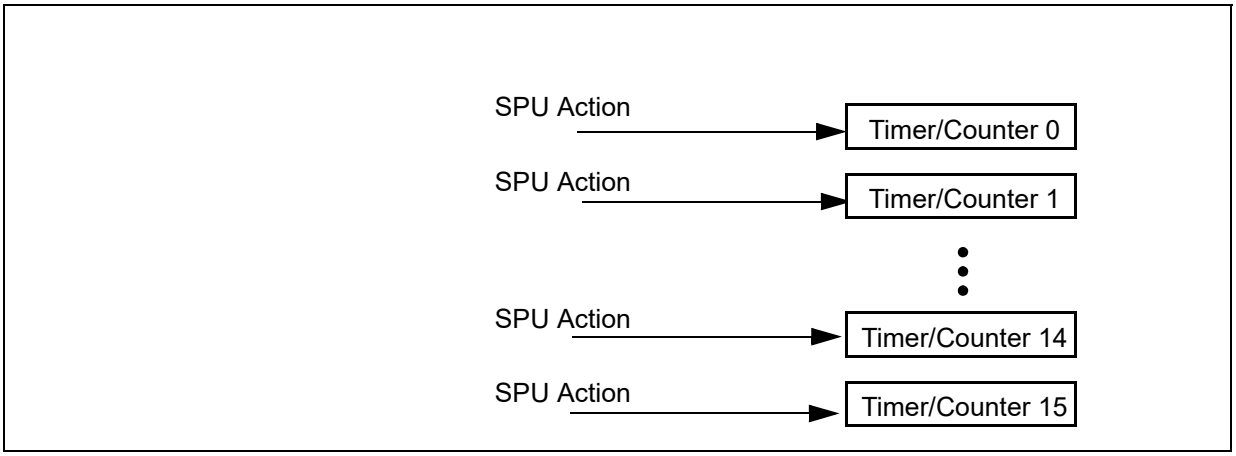
63.6.5.2 Counter input selection at SPU level

Counter increment inputs derive from the SPU action. These inputs include signals from the primary SPU muxes or SPU events of counters.

63.6.5.3 Counters increment assignment

When configured as a counter, the counter has an increment signal. The assignment of which signal is used to increment a counter is handled in the SPU, and is configurable by user. The assignment of increment signals to counters is shown in *Figure 1117*.

Figure 1117. SPU counters increment assignment



63.6.6 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 63.2: SPU actions*. 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.

63.6.7 Optional trace group

User defined trace group can be defined through the configuration register. Configuration registers, defined in *Table 1049*, are available where the user can group the different traces from same or different clients.

63.6.8 Format for CKSRC & CKDATA Values

The format of the CKSRC is defined for the SPU Peripheral (as In-Circuit trace client):

Table 1057. CKSRC format

ICT CLIENTS	CKSRC Value
SPU Counter Trace Client	b00_0000
Reserved	b00_0001 -- b11_1111

The format for the CKDATA field for the SPU Counter trace client is:

Table 1058. CKDATA format

CKDATA	
CKDATA[CTR_Index]	CKDATA[CTR_VALUE]
4 bits	32 bits

64 Development Trigger Semaphore (DTS)

64.1 Introduction

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways this module can be used, including as a component of an external real-time data acquisition system.

Note: When used as a component of a triggered data acquisition system, Nexus read/write access is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2011 Nexus standards, which use the Nexus Auxiliary port.

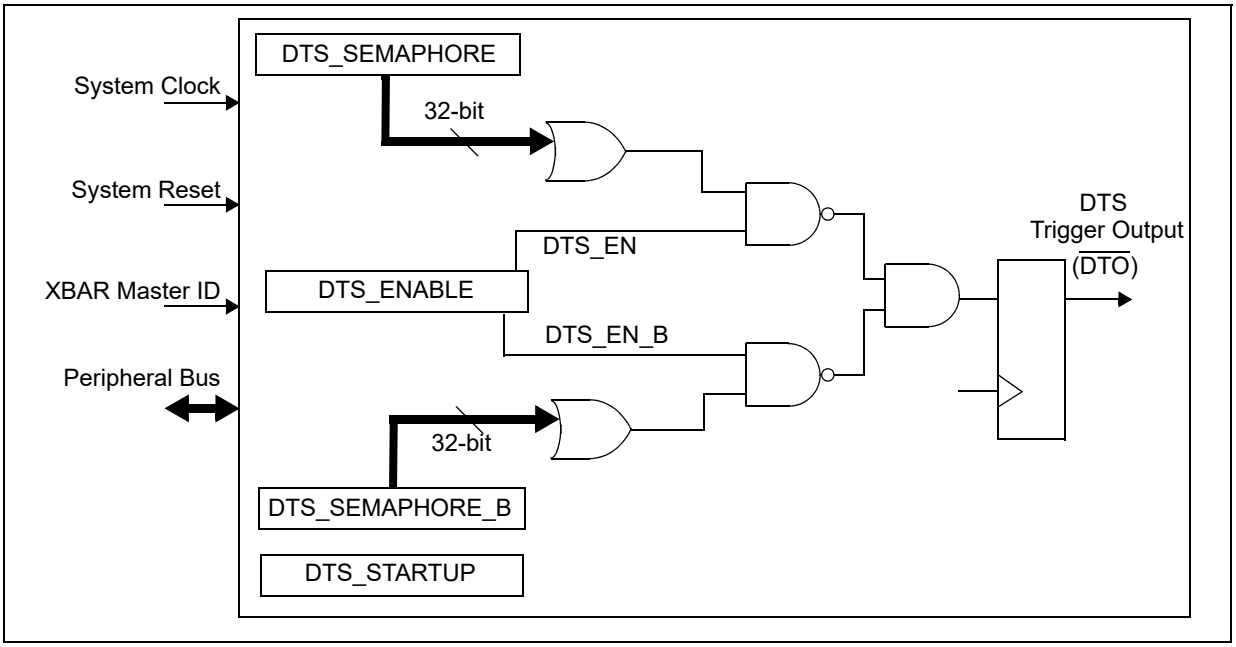
64.2 Overview

The Development Trigger Semaphore (DTS) module consists of registers and a small amount of combinational logic to generate an output signal—DTS Trigger Output ($\overline{\text{DTO}}$). The registers are as follows:

- DTS_SEMAPHORE register—Any bit in this 32-bit register, when set to a value of logic ‘1’, causes the DTS module output signal to be asserted if the corresponding DTS_EN bit is set in the DTS_ENABLE register. This enables an external tool to detect up to 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access.
- DTS_SEMAPHORE_B register—Any bit in this 32-bit register, when set to a value of logic ‘1’, causes the DTS module output signal to be asserted if the corresponding DTS_EN_B bit is set in the DTS_ENABLE register. This enables an external tool to detect an additional 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access.
- DTS_STARTUP register—This register provides a mechanism for the external tool to notify software running on the CPU that the tool is connected and can provide information about either the type of tool or options that can be used by the software.
- DTS_ENABLE register—This register provides an enable/disable capability for the DTS feature.

The architecture is shown in *Figure 1118*.

Figure 1118. DTS block diagram

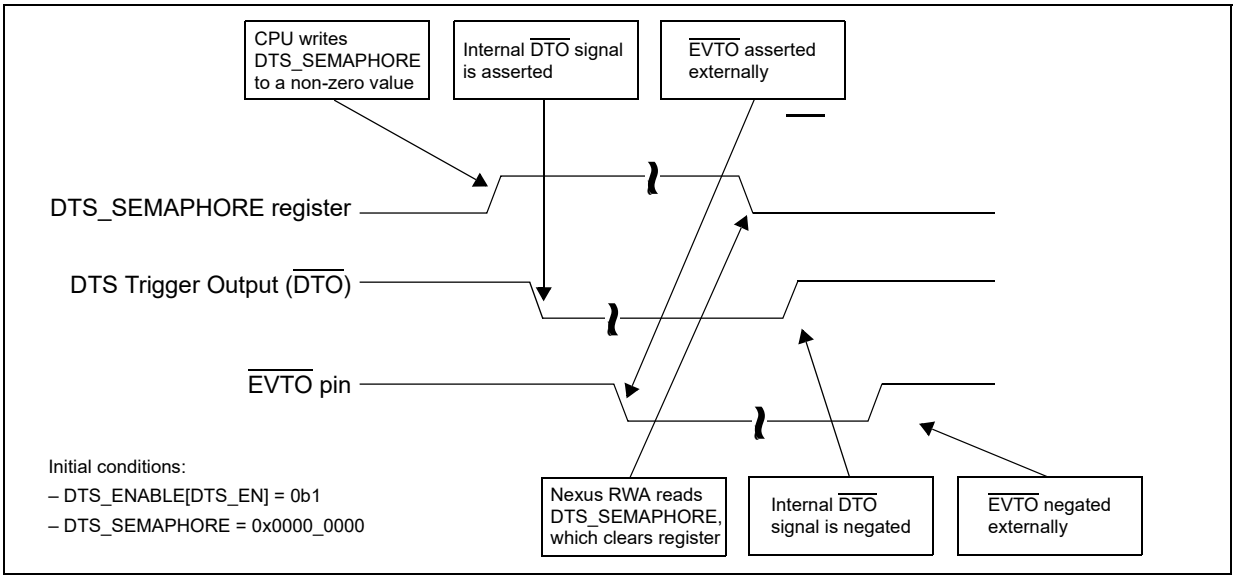


The DTS Trigger Output ($\overline{\text{DTO}}$) signal is connected to one of the $\overline{\text{EVTO}}$ inputs of the Debug and Calibration Interface (DCI). The other $\overline{\text{EVTO}}$ inputs are connected to the other Nexus modules in the device. $\overline{\text{DTO}}$ is asserted when any bit in a semaphore register is set and the DTS function of that register is enabled.

Note: When the DTS module is enabled (DTS_ENABLE[DTS_EN] = 0b1 or DTS_ENABLE[DTS_EN_B] = 0b1), all other Nexus EVTO functions associated should be disabled by the tool and $\overline{\text{EVTO}}$ becomes the $\overline{\text{DTO}}$. Unlike the $\overline{\text{EVTO}}$ function that only asserts for one clock, the $\overline{\text{DTO}}$ function remains asserted until the tool reads the semaphore register that is the source of the assertion, clearing the register's contents.

Figure 1119 shows an example of the chain of events that begins with setting of a bit in the DTS_SEMPAHORE register and the clearing of the register caused by a Nexus read.

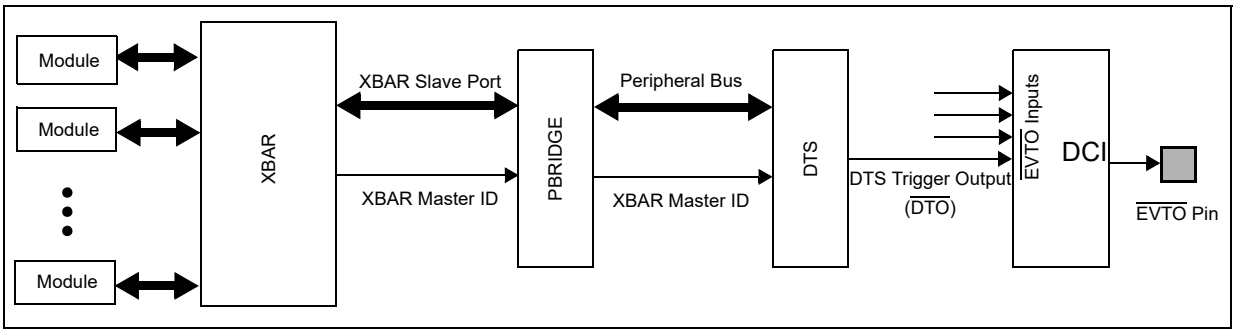
Figure 1119. $\overline{\text{DTO}}$ event sequence example



64.3 DTS device connections

Figure 1120 shows the DTS device connections. The DTS module connects to the Peripheral Bridge (PBRIDGE) for access to the registers. The PBRIDGE is connected to the device modules through a slave port of the Crossbar bus interface (XBAR).

Figure 1120. DTS device connections



The registers have limited access as described in *Section 64.3.1: DTS register access*. Access is based on the XBAR Master ID of the accessing module. Writing to the semaphore registers is limited to the cores and the eDMA module and is restricted to only setting bits. Only an access via a Nexus Read/Write Access from an external tool can clear bits in the semaphore registers (semaphore register bits are cleared automatically when read via Nexus/Write Access). Similarly, the DTS_ENABLE and DTS_STARTUP registers can only be written via a Nexus Read/Write Access.

Note: Nexus Read/Write Accesses use the core load/store bus to perform accesses, but Nexus accesses have a different Master ID than normal core load/stores.

64.3.1 DTS register access

A summary of accesses to all DTS registers by bus masters is provided in *Table 1059*. Only proper 32-bit accesses are valid. The effects of write accesses that are not 32 bits are not defined.

Table 1059. DTS register access effects

Register	32-bit read				32-bit write			
	RWA(1)	Core	eDMA	FlexRay(2)	RWA(1)	Core	eDMA	FlexRay(2)
DTS_ENABLE	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_STARTUP	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_SEMAPHORE	Data and Clear(3)	Data	Data	Data	No effect	Bit OR	Bit OR	No effect
DTS_SEMAPHORE_B	Data and Clear(3)	Data	Data	Data	No effect	Bit OR	Bit OR	No effect

1. Nexus Read/Write access via an external tool
2. FlexRay or other master
3. A read of the semaphore registers by Nexus Read/Write Access module is destructive and clears all bits in the register

Access to DTS module registers is controlled based on the XBAR Master ID of the accessing module.

Note: The XBAR Master ID should not be confused with the Master Port number of the XBAR. See Chapter 14: Crossbar Switch (XBAR), for details.

Tools must access the DTS registers through the Nexus Read/Write Access mechanism of a core. External tool accesses through either core appear as if the access is via the core and therefore do not have the same level of access as a Nexus Read/Write Access.

64.4 Memory map and register definition

Table 1060 shows the memory map of the DTS module registers.

Table 1060. DTS memory map

Offset	Register	Access	Reset Value	Section/Page
0x0	ENABLE—DTS Output Enable Register	Restricted R/W(1)	0x0000_0000	*64.4.1.1/1830*
0x4	STARTUP—DTS Startup Register	Restricted R/W(1)	0x0000_0000	*64.4.1.2/1830*
0x8	SEMAPHORE—DTS Semaphore Register	Restricted R/W(1)	0xFFFF_FFFF	*64.4.1.3/1831*
0xC	SEMAPHORE_B—DTS Semaphore Extension	Restricted R/W(1)	0xFFFF_FFFF	*64.4.1.4/1832*

1. Only certain types of accesses are allowed. See the register figure note of each register for a description of the accesses that are allowed.

64.4.1 Register descriptions

64.4.1.1 DTS output enable register (ENABLE)

The DTS_ENABLE register controls the DTS Trigger Output ($\overline{\text{DTO}}$) and whether $\overline{\text{DTO}}$ is active on the $\overline{\text{EVTO}}$ output pin of the device. *Figure 1121* shows the format of the DTS_ENABLE register.

Note: Access to the DTS_SEMAPHORE, DTS_SEMAPHORE_B, and DTS_STARTUP registers are unaffected by the state of this register.

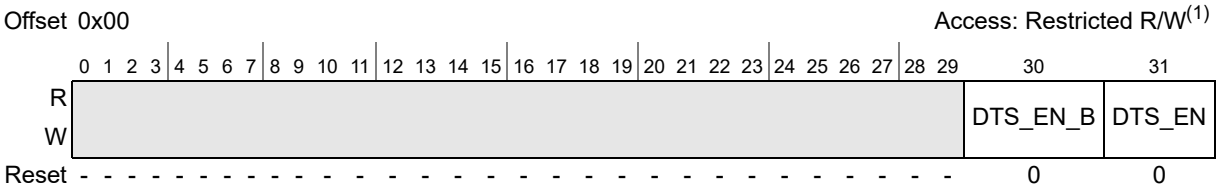


Figure 1121. DTS_ENABLE register

1. The DTS_ENABLE register can be read by the core, but can only be written by a Nexus Read Write Access (RWA).

The DTS_ENABLE register is described in *Table 1061*.

Table 1061. DTS_ENABLE register field descriptions

Field	Description
DTS_EN	DTS Enable. Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).
DTS_EN_B	DTS Enable B. Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE_B register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE_B register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).

64.4.1.2 DTS startup register (STARTUP)

Figure 1122 shows the format of the DTS_STARTUP register, used for tool detection and startup information exchange between external data acquisition tool and the embedded controller.

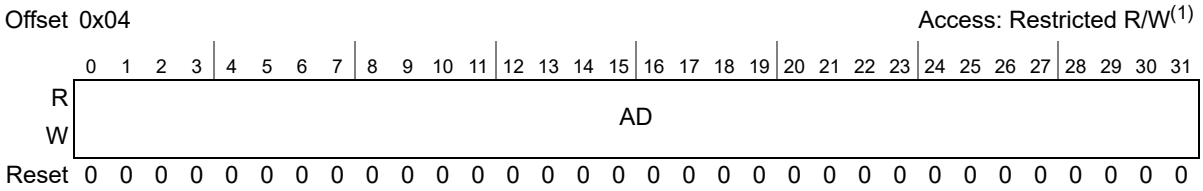


Figure 1122. DTS_STARTUP register

1. The DTS_STARTUP register can be read by the core, the eDMA module and Nexus but can only be updated by a Nexus Read Write Access (RWA).

The DTS_STARTUP register is described in *Table 1062*.

Table 1062. DTS_STARTUP register field descriptions

Field	Description
AD	Application Dependent register bits. The bits have no defined meaning to the microcontroller. They are used by an external tool to pass information (for example, application options and status) to application software running on target microcontroller at startup time. Use a Nexus RWA 32-bit write access to update the contents of this register. A device reset (either from the RESET pin or an internally generated reset) clears all bits in the register. A Nexus reset does not change the contents of the register.

64.4.1.3 DTS semaphore register (SEMAPHORE)

Figure 1123 shows the format of the DTS_SEMAPHORE register. The DTS_SEMAPHORE register is used by software to assert the DTO signal on the device EVTO pin. A one in any bit of this register causes the DTO signal on the EVTO pin to be driven low. The intended use of this register is for the DTO signal to notify tools that data is available. Individual bits are used to identify the specific data.

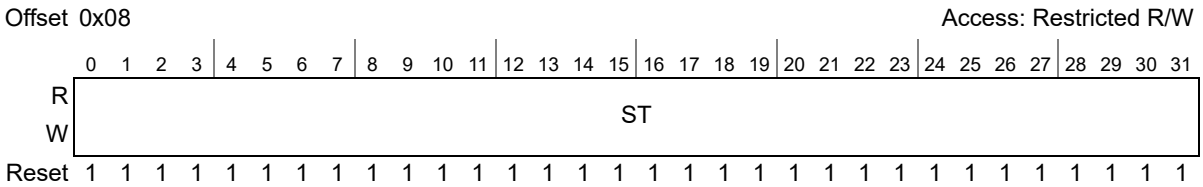


Figure 1123. DTS_SEMAPHORE register

The DTS_SEMAPHORE register is described in *Table 1063*.

Table 1063. DTS_SEMAPHORE register field descriptions

Field	Description
ST	Semaphore Trigger. When a core or eDMA writes a logical ‘1’ to a bit, the bit is set. A write of ‘0’ by the core or DMA does not change the state of the bit. – All register bits are set to ‘1’ by a device reset. – A Nexus reset does not change the state of this register. – The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA. – For the core or DMA, only 32-bit write or read accesses are valid. – A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

64.4.1.4 DTS semaphore extension (SEMAPHORE_B)

Figure 1124 shows the format of the DTS_SEMAPHORE_B register. The DTS_SEMAPHORE_B register is used by software to assert the DTO signal on the device EVTO pin. A one in any bit of this register causes the DTO signal on the EVTO pin to be driven low. The intended use of this register is for the DTO signal to notify tools that data is available. Individual bits are used to identify the specific data.

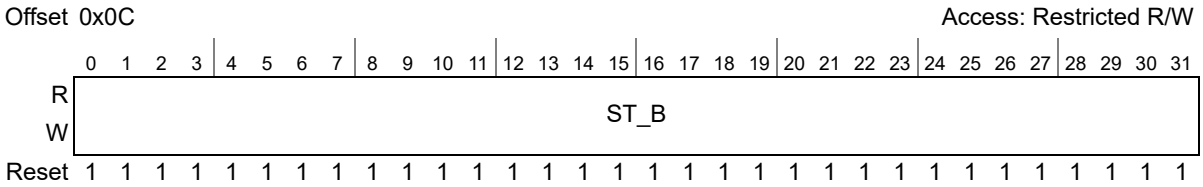


Figure 1124. DTS_SEMAPHORE_B register

The DTS_SEMAPHORE_B register is described in *Table 1064*.

Table 1064. DTS_SEMAPHORE_B register field descriptions

Field	Description
ST_B	Semaphore Trigger Extension. When a core or eDMA writes a logical ‘1’ to a bit, the bit is set. A write of ‘0’ by the core or DMA does not change the state of the bit. – All register bits are set to ‘1’ by a device reset. – A Nexus reset does not change the state of this register. – The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA. – For the core or DMA, only 32-bit write or read accesses are valid. – A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

64.5 Example application

The calibration process of a new engine requires real-time access to calibration tables and the ability to update the tables in real-time The DTS module enables this capability by

enabling software to assert a signal to an external device pin to notify an external tool that data is available. The tool can then retrieve the data.

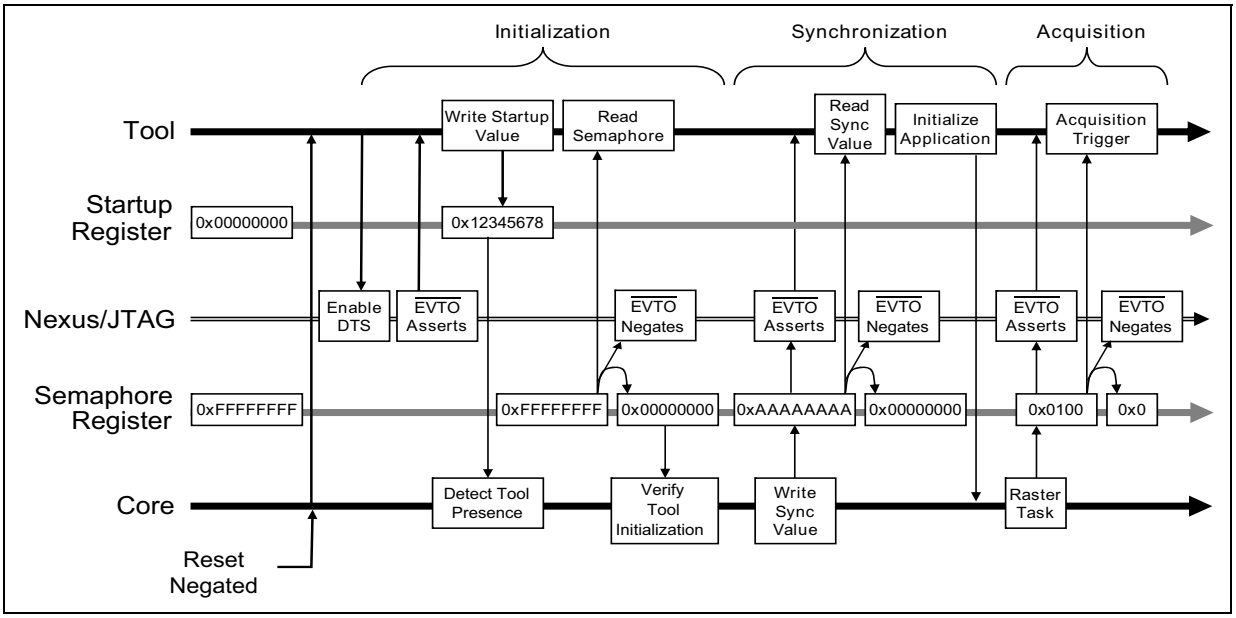
In this type of application the DTS_SEMAPHORE register and DTS Trigger Output ($\overline{\text{DTO}}$) signal provide a mechanism to notify the calibration tool that the calibration variable or variables (or sets of measurements), up to 32, have been updated with new values and are available for the tool to access.

Note: It is the user's responsibility to ensure that the tool has time to retrieve the data prior to that particular trigger being set a second time. It is also permissible to have multiple triggers active at the same time or for a second trigger to be set before a previous trigger has been serviced, as long as it is not the same trigger (unless it is acceptable to the tool to not receive every data set).

Figure 1125 shows an example DTS startup sequence for an external real-time data acquisition system. The startup and synchronization sequence can be as simple or as complicated as the need requires. However, a typical startup sequence is as follows:

1. The DTS_STARTUP register is cleared by a power on reset or any CPU reset.
2. The tool writes a non-zero value to the DTS_STARTUP register.
3. The CPU (user application software) then reads the value of the DTS_STARTUP register. Based on this value, different initialization options can be selected. The bits can be used for any application specific definitions.
4. Since the DTS_SEMAPHORE register is cleared when the tool reads the current value. The tool should perform all necessary initialization before reading this register. The application software can then check that the DTS_SEMAPHORE register was cleared by the tool, to determine that it is safe to start using it for its intended raster trigger semaphore function.
5. An optional hand shake from the CPU can be used to inform the tool that the user software has detected that the tool is attached and the CPU has performed the proper initialization for the tool by writing a predefined value to the DTS_SEMAPHORE register (the example shown in the figure above uses 0xAAAA_AAAA—all A's was used since it is unrealistic that 16 channels could be enabled very quickly after start up after a reset).

Figure 1125. DTS startup sequence example



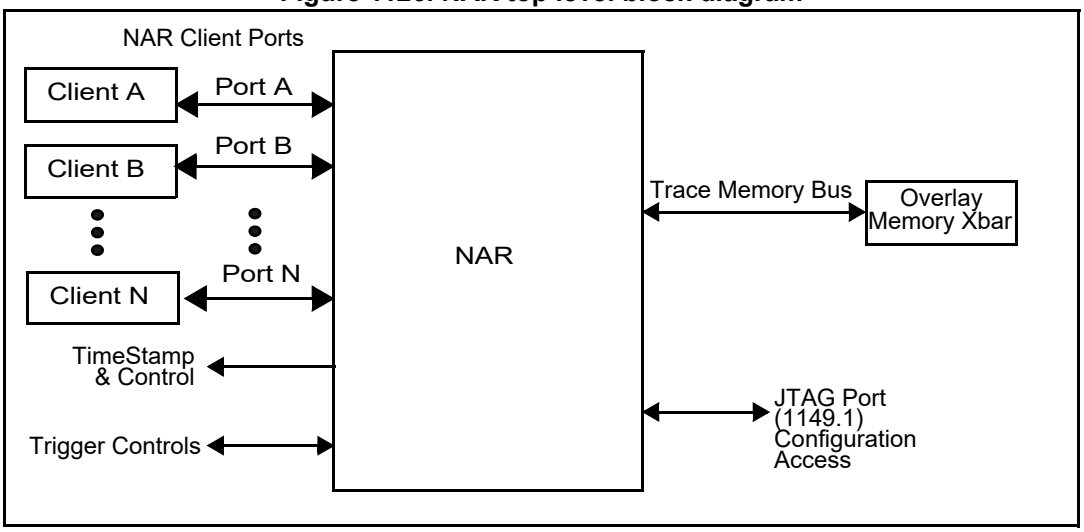
65 Nexus Aurora Router (NAR)

65.1 Introduction

On the device, there are multiple blocks that require development interface support. The blocks that are Nexus-compliant (based on the IEEE-ISTO 5001 standard) are expected to interface with the development tool through a dedicated high-bandwidth debug port. The Nexus Aurora Router (NAR) controls the usage of the 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.

Top-level architecture is shown in *Figure 1126*. Multiple clients send messages by sharing the high-bandwidth port through the NAR. For a list of the specific clients that are connected to the NAR, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Figure 1126. NAR top level block diagram



The trace memory bus port provides added flexibility to the channel trace data to memory on the Nexus Aurora crossbar. Setup and configuration of the NAR is performed by accessing the configuration registers via a JTAG (1149.1) port.

Each Nexus client has a dedicated receive queue to allow seamless switching between sources at Nexus message boundaries. Incoming Nexus data is parsed into the receive queues so that end-of-message can be determined. Messages are then transferred into the main queue. An arbitration block determines the order in which clients are serviced, using an optional client request level bus to prioritize the data (lacking request level data, a simple round-robin algorithm is used). The NAR has two output ports: the high bandwidth debug port (Aurora out) and the trace memory bus. If both output ports are enabled at the same time, the main queue is logically partitioned inside NAR. The Production Device (PD) does not support partitioning of the main queue; therefore, only one of the ports can be active at a time. Data going through the Aurora out port or the Trace port goes through an additional translation block to put the data in the correct format. The actual Aurora formatting is done in the NAL. Additional blocks for message generation and timestamp generation are described in more detail in the following sections.

The NAR supports the following client interfaces:

- Legacy interface to support existing client interface protocol through a conversion gasket logic with clock domain transfer
- New client interface with clock domain transfer

65.2 Overview

The NAR serves as an arbiter for the high-bandwidth debug port (HBDP). The NAR pulls Nexus messages from multiple clients, queues the data, and forwards it (based on the user-controlled configuration) to the high-speed Aurora Out port and/or the Trace port. The trace memory bus controller allows the NAR to transmit data via the normal system interface. NAR control and configuration registers are accessed via JTAG (1149.1) port.

65.3 Features

The NAR supports the following features:

- Connects 2 client(s) with dedicated receive queue(s). Clients can transmit data as long as there is space in its associated receive queue, but only the active port receive queue empties into the main queue.
- Legacy client interface for backward compatibility
- Aurora output data format for dedicated high-bandwidth serial transmission
- Aurora output format support at client interface
- Trace Memory port (64-bit address and data) can be used to write Nexus messages directly to external overlay memory. Block transfer mode provides highest-bandwidth option.
- Configurable Start Address and Range Selection for Trace Memory port write operation provides flexibility to choose Overlay memory Section and range to store Trace data.
- Time-stamping provides a central synchronization resource for message ordering. Gray coded timer output for clients working in different clock domains.
- Re-allocatable receive queues allow queue-space of an inactive client to be reassigned to an active client
- Stall detection and handling allows NAR to continue when a client stops sending data in the middle of a message
- Event generation on External Trace Memory (overlay) is programmable
- Suppress mode can be used to shut down message delivery and relieve back-pressure to clients without turning off Nexus messaging in the clients.

65.4 Register definition

The NAR configuration registers provide configuration and control for all NAR operations. They are accessible through the JTAG (1149.1) port. Addressing (within the local NAR address space) for the NAR registers is shown in the following table. If the NAR clock is turned off, the NAR registers cannot be accessed.

Table 1065. NAR registers

Register name	Read/ Write	JTAG index (NAR relative)	Location
NAR_CR—NAR Control Register	R/W	0	*on page 1837*
NAR_ST—NAR Status Register	R	1	*on page 1839*
NAR_SFR—Source Filter Register	R/W	2	*on page 1841*
NAR_TFR—Type Filter Register	R/W	3	*on page 1842*
NAR_TBALO—Trace memory port base address, low	R/W	4	*on page 1843*
NAR_TBAHI—Trace memory port base address, high	R/W	5	*on page 1843*
NAR_TCR—Trace Memory bus control register	R/W	6	*on page 1844*
NAR_STCR—Suppress Trigger Control Registers 1–4	R/W	7 (Trigger 1) 8 (Trigger 2) 9 (Trigger 3) A (Trigger 4)	*on page 1845*
NAR_CSSR–Client Suppress Status Register	R	B	*on page 1847*
NAR_CDR—Client Disable Register	R/W	C	*on page 1847*
NAR_AHFPAR—Trace memory fill level and partition configuration	R/W	E	*on page 1849*

65.4.1 Register descriptions

65.4.1.1 NAR control register (NAR_CR)

The NAR control register provides all the main control and enable bits for the NAR and its output ports. The following figure shows the format of the NAR_CR.

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NCD	TEN	TRS	TSG		SED	SRD	TBW	AQP		0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WPE			0	0	0	0	0	ALT		0	AFA	0	0	PUS	NSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1127. NAR control register (NAR_CR)

The NAR_CR is described in the following table.

Table 1066. NAR_CR field descriptions

Field	Description
31 NEN	NAR Enable. 0 NAR is disabled unless HBDP is independently enabled. See *Section 65.5.2.2, Disabled* for more information. 1 NAR is enabled.
30 NCD	NAR Communication Disable. 0 NAR monitors clients for activity 1 NAR communication with clients is disabled
29 TEN	Timestamp Enable. 0 Timestamp packet is not included in messages 1 Timestamp packet is included in messages from all client including NAR generated messages
28 TRS	Timestamp Reset. 0 to 1 transition: reset timestamp counter in NAR
27–26 TSG	Timestamp Granularity (Mode). **Note:** Some clients may only support timestamps on every message. 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 32nd message
25 SED	Stall Error Disable. 0 NAR injects dummy complete message when active client stalls 1 NAR does not inject dummy complete message
24 SRD	Stall Recovery Disable. 0 Stalled receive queue resumes operation when its client comes back on line 1 Stall error recovery is disabled
23 TBW	Tracebuffer Wrap. 0 NAR does not capture new data when main queue is full 1 NAR begins overwriting main queue from the beginning upon queue overflow
22–21 AQP	Alternate Queue Partition. 00 Queue partitioning is disabled 01 Reserved 10 Reserved 11 Reserved
20	Reserved and must be written with 0
19–16	Reserved and must be written with 0s
15–13 WPE	Watchpoint Enable 000 NAR watchpoint messaging is disabled xx1 NAR generates a watchpoint message upon a sync event x1x NAR generates a watchpoint message upon a trace memory bus or tracebuffer block full/wrap event 1xx NAR generates a watchpoint message upon a suppress-mode event
12	Reserved
11	Reserved. This bit must be written with 0.
10–9	Reserved and must be written with 0s.

Table 1066. NAR_CR field descriptions(Continued)

Field	Description
8	Reserved
7–6 ALT	Alternate Port. Defines which port is the target for non-filtered messages; main port in single-port applications. 00 No alternate port. Running in trace-buffer mode. 01 Aurora out is the ALT 10 Trace memory bus is the ALT 11 Reserved
5	Reserved
4 AFA	Automatic Flush Accumulators. 0 Output queue accumulators do not flush stranded beats 1 Accumulators automatically flush stranded beats
3	Reserved. This bit must be written with 0.
2	Reserved and must be written with 0.
1 PUS	Power-Up Suppressed. 0 NAR powers up active 1 NAR powers up in global suppressed mode
0 NSR	NAR Soft Reset. 1 Resets all configuration registers and sets queue address pointers to the top of their respective queues. Also resets itself after a single-cycle delay. (This does not affect other blocks)

65.4.1.2 NAR status register (NAR_ST)

The NAR_ST register provides information on NAR activity and queue status, as well as HBDP status and errors. It is a read-only register. The following figure shows the format of the NAR_ST.

Register index: 1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NUM			NCE				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HPR	HPE	MPE	MOV	0	TOV	QWA									
W		w1c	w1c	w1c		w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 1128. NAR status register (NAR_ST)

The NAR_ST register is described in the following table.

Table 1067. NAR_ST field descriptions

Field	Description
31 NEN	NAR Enabled. 0 NAR is disabled and inactive 1 NAR is enabled with clocks on
30–28 NUM	NAR Mode. 000 NAR is in disabled mode 001 NAR is in global suppressed mode 010 NAR is in communication-disabled mode 011 NAR is in tracebuffer mode 100 NAR is in active transmit mode 101 Reserved 110 Reserved 111 NAR is in run-control mode
27–24 NCE	NAR Config Error. See *Table 1081* for coding
23–16	Reserved
15 HPR	HBDP Ready. 0 HBDP (Aurora Out port) is not ready 1 HBDP is ready and accepts data for transmission
14 HPE	HBDP Error. 0 No HBDP error 1 Error condition exists on HBDP
13 MPE	Trace memory bus Error. 0 No Trace memory bus error 1 Error condition detected on trace transaction
12 MOV	Trace memory bus block overflow. 0 No Trace memory bus overflow 1 Targeted external memory block is full. Trace memory bus writes are halted (or wrapped).
11	Reserved
10 TOV	Tracebuffer Overflow. 0 No tracebuffer overflow 1 The tracebuffer has overflowed/wrapped at least once
9–0 QWA	Queue Write Address. Current value of the output queue write pointer (last valid write + 1)

65.4.2 Message filtering registers

The two output ports (HBDP and trace memory bus) are independently controllable. In the BD NAR, two ports can be active simultaneously. When multiple ports are active, user-programmable filters are determine which port receives a message, based on the message source or type. Filters apply to the designated Message Select Port (MSP), and are used to select messages for that port. All other messages go to the other (alternate) port. If multiple ports are used, an MSP must be designated; otherwise a configuration error results. See *Section 65.5.6, Output arbitration*, for details on how the NAR arbitrates between multiple output ports. These registers should only be set in the BD NAR because partitioning is not supported in the PD NAR.

65.4.2.1 Source filter register (NAR_SFR)

The MSP can be configured to accept or exclude messages from up to four sources. The sources which are to be filtered are programmed in the source filter register (NAR_SFR) as shown in the following figure. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 2

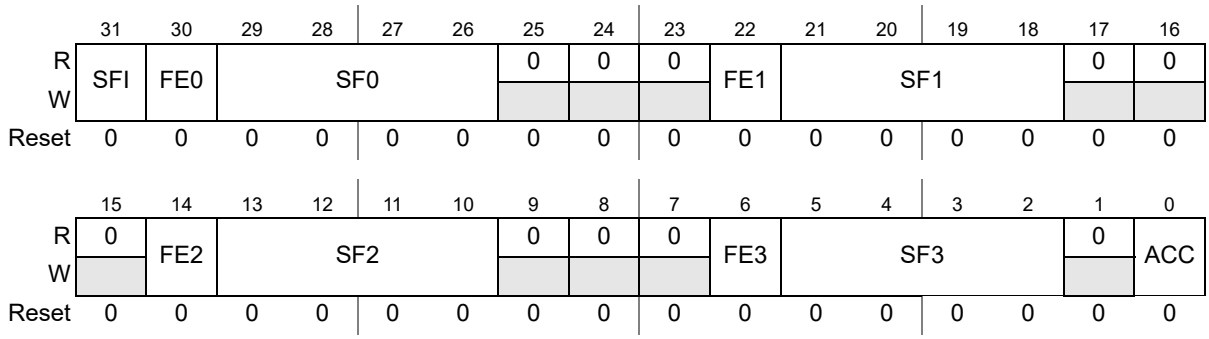


Figure 1129. Source filter register (NAR_SFR)

The NAR_SFR is described in the following table. The source trace IDs, that are compared to the source filter fields (SFn), are device specific. For a list of the clients that are connected to the NAR and their corresponding Nexus source trace IDs, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 1068. NAR_SFR field descriptions

Field	Description
31 SFI	Source Filter Invert. 0 Match if message source equals any of the enabled filters 1 Match if message source equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. SF0 ignored 1 Filter 0 enabled and contained in SF0
29–26 SF0	Source Filter 0. The message source trace ID is compared with this field. See *Section 65.5.6, Output arbitration* for an example.
25–23	Reserved.
22 FE1	Enable Filter 1. 0 Filter 1 disabled. SF1 ignored 1 Filter 1 enabled and contained in SF1
21–18 SF1	Source Filter 1. The message source trace ID is compared with this field. See *Section 65.5.6, Output arbitration* for an example.
17–15	Reserved
14 FE2	Enable Filter 2. 0 Filter 2 disabled. SF2 ignored 1 Filter 2 enabled and contained in SF2
13–10 SF2	Source Filter 2. The message source trace ID is compared with this field. See *Section 65.5.6, Output arbitration* for an example.

Table 1068. NAR_SFR field descriptions(Continued)

Field	Description
9–7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. SF3 ignored 1 Filter 3 enabled and contained in SF3
5–2 SF3	Source Filter 3. The message source trace ID is compared with this field. See *Section 65.5.6, Output arbitration* for an example.
1	Reserved
0 ACC	OR source and type filters. 0 Match if source AND type filters match 1 Match if source OR type filters match

65.4.2.2 Message type filter register (NAR_TFR)

The MSP can be configured to accept or exclude messages of up to four types, as encoded in the message TCODE field. The TCODEs which are to be filtered are programmed in the message type filter register (NAR_TFR) as shown in the following figure. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 3

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TFI	FE0	TF0						0	FE1	TF1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	TF2						0	FE3	TF3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1130. Message type filter register (NAR_TFR)

The NAR_TFR is described in the following table.

Table 1069. NAR_TFR field descriptions

Field	Description
31 TFI	Type Filter Invert. 0 Match if message TCODE equals any of the enabled filters 1 Match if message TCODE equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. TF0 ignored 1 Filter 0 enabled and contained in TF0
29–24 TF0	Type Filter 0. The message TCODE is compared with this field. See *Section 65.5.6, Output arbitration* for an example.
23	Reserved

Table 1069. NAR_TFR field descriptions(Continued)

Field	Description
22 FE1	Enable Filter 1. 0 Filter 1 disabled. TF1 ignored 1 Filter 1 enabled and contained in TF1
21–16 TF1	Type Filter 1. The message TCODE is compared with this field. See *Section 65.5.6, Output arbitration* for an example.
15	Reserved
16 FE2	Enable Filter 2. 0 Filter 2 disabled. TF2 ignored 1 Filter 2 enabled and contained in TF2
13–8 TF2	Type Filter 2. The message TCODE is compared with this field. See *Section 65.5.6, Output arbitration* for an example.
7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. TF3 ignored 1 Filter 3 enabled and contained in TF3
5–0 SF3	Type Filter 3. The message TCODE is compared with this field. See *Section 65.5.6, Output arbitration* for an example.

65.4.3 Trace memory bus configuration registers

The trace memory bus can be used to transfer debug messages out to a dedicated block of external memory space. The trace memory bus configuration registers are used to specify the location and size of the external debug space.

65.4.3.1 Trace memory bus base address registers (NAR_TBALO and NAR_TBAHI)

The trace memory bus base address registers determine where Nexus data sent out through the trace memory bus goes. The NAR sends out blocks of data on the trace memory bus, writing to an address block defined by the specified base address. Data can be transmitted either on an as-available, word-wide basis, or in larger blocks. In block transfer mode, the size of the data block is determined by the BTM field of the Control Register; the default is 64 bytes. Even if block transfer mode is not used, the 5, 6, 7, or 8 (depending on the BTS field in NAR_TCR register) LSBs of the base address are not accessible. This forces the debug memory space to be aligned to a block boundary.

When read, the address registers return the current address, allowing the user to monitor the progress of a trace memory transfer. The following figures show the format of the trace memory base address registers (NAR_TBALO and NAR_TBAHI). For a description of the allowed address for these registers, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Register index: 5

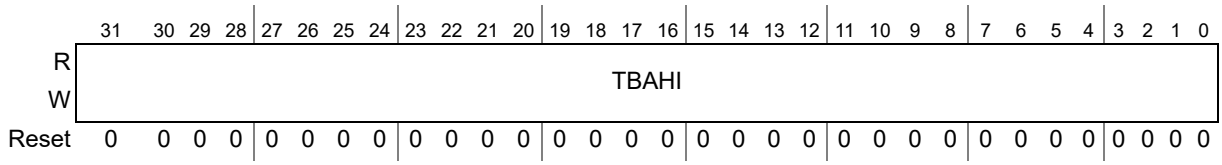


Figure 1131. Trace memory base address register NAR_TBAHI

Register index: 4

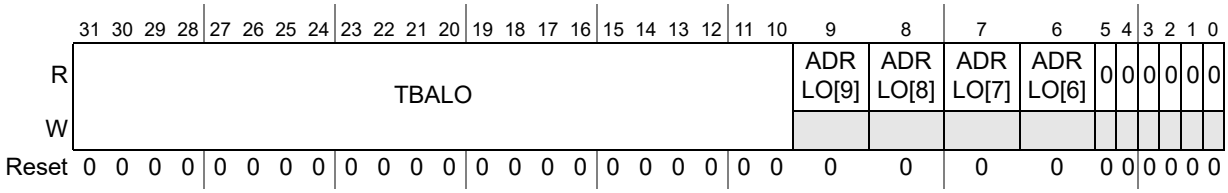


Figure 1132. Trace memory base address register NAR_TBALO

The NAR_TBALO and NAR_TBAHI registers are described in the following tables.

Table 1070. NAR_TBAHI field descriptions

Field	Description
31–0 TBAHI	Trace Debug Memory Base Address on AHB-EW port, upper part. For 32-bit address applications this field is all zeros.

Table 1071. NAR_TBALO field descriptions

Field	Description
31–10 TBALO	Trace Debug Memory Base Address on AHB-EW port, lower part
9 ADRLO[9]	Base Address LSB for BTS = 11 (384-byte block)
8 ADRLO[8]	Base Address LSB for BTS = 10 (256-byte block)
7 ADRLO[7]	Base Address LSB for BTS = 01 (128-byte block)
6 ADRLO[6]	Base Address LSB for BTS = 00 (64-byte block)
5–0	Reserved

65.4.3.2 Trace memory bus control register (NAR_TCR)

The trace memory bus control register is used to control the attributes of the trace memory bus port, such as block transfer size and message priority. Since the trace memory bus transfers are expected to go to a static block of external memory, a maximum transfer size must be specified to prevent the NAR from overwriting memory it is not authorized to access. This is done via the MXFR max transfer size register field. The memory allocation is

specified in units of the trace memory bus block transfer size (64, 128, 256, or 384 bytes), so the actual size of the reserved memory block depends on the BTS setting. The following figure shows the format of the NAR_TCR.

Register index: 6

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BTM	BTS			BTH			0	0	0	TBW	MXFR				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MXFR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1133. Trace memory bus control register (NAR_TCR)

The NAR_TCR is described in the following table.

Table 1072. NAR_TCR field description

Field	Description
31 BTM	Block Transfer Mode. 0 Data for the Trace Memory bus port is transmitted one word at a time, upon first availability 1 Data to the Trace Memory bus port is bundled into blocks before transmitting
30–29 BTS	Block Transfer Size. 00 Block transfer size = 64 bytes 01 Block transfer size = 128 bytes 10 Block transfer size = 256 bytes 11 Block transfer size = 384 bytes
28–27 BTH	Block Transfer Threshold. 00 Full threshold. Number of entries in partition must equal block transfer size before block transfer starts 01 Begin transfer when block is 25% full 10 Begin transfer when block is 50% full 11 Begin transfer when block is 75% full
26–24	Reserved. These bits must be written with 0.
32 TBW	Trace Memory Bus Block Wrap. 0 NAR stops writing to the Trace Memory bus debug memory space once Max Size has been reached 1 NAR overwrites Trace Memory bus debug memory from the beginning upon Trace Memory bus block overflow.
22–0 MXFR	Trace Memory Bus Max Transfer Size. Total size of the dedicated debug memory space, in Trace Memory bus data transfer blocks.

65.4.4 Suppress mode triggers register (NAR_STCR)

Four suppress mode triggers are provided for precise control of the NAR suppress mode feature described in *Section 65.5.4, Suppress mode*. The suppress mode triggers are only available on the PD NAR and do not apply to the BD NAR. The registers are present in the BD NAR, but input suppress triggers are not connected. Each trigger can be programmed to

suppress or wake clients individually or the entire NAR. All receive queues can be suppressed on the occurrence of a selected trigger, or the full NAR can be brought out of suppress mode upon the occurrence of the appropriate trigger. A trigger can also be programmed to toggle one or more clients between the suppressed and active states. Each Suppress Trigger Control Register, shown in the following figure, is associated with one of the triggers, and are used to determine the action taken, if any, upon the assertion of that trigger. The suppress triggers are connected to the Sequence Processing Unit (SPU) action unit.

Register index: 7–A

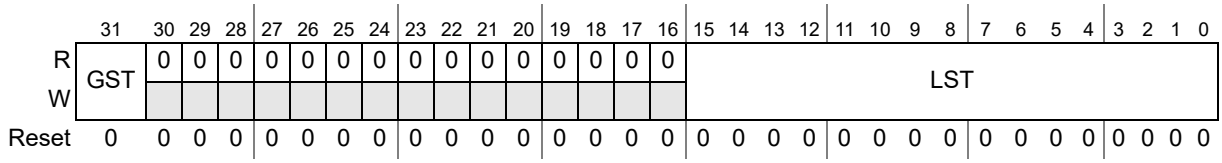


Figure 1134. Suppress trigger control registers (NAR_STCR1–4)

The NAR_STCR are described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress trigger, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 1073. NAR_STCR field descriptions

Field	Description
31 GST	Global Suppress Trigger. 0 The associated trigger cannot cause NAR to enter global suppress mode 1 The associated trigger is enabled for global suppress mode
29–16	Reserved
15–0 LST	Local Suppress Trigger. 0000000000000000The associated trigger is disabled for local suppress mode for all clients xxxxxxxxxxxxxxx1Trigger is enabled for Client 1 local suppress xxxxxxxxxxxxxx1xTrigger is enabled for Client 2 local suppress xxxxxxxxxxxxx1xxTrigger is enabled for Client 3 local suppress xxxxxxxxxxxx1xxxTrigger is enabled for Client 4 local suppress xxxxxxxxxxx1xxxxTrigger is enabled for Client 5 local suppress xxxxxxxxxx1xxxxxTrigger is enabled for Client 6 local suppress xxxxxxxxx1xxxxxxTrigger is enabled for Client 7 local suppress xxxxxxxx1xxxxxxxTrigger is enabled for Client 8 local suppress xxxxxxx1xxxxxxxxTrigger is enabled for Client 9 local suppress xxxxxx1xxxxxxxxxTrigger is enabled for Client 10 local suppress xxxxx1xxxxxxxxxxTrigger is enabled for Client 11 local suppress xxxx1xxxxxxxxxxxTrigger is enabled for Client 12 local suppress xxx1xxxxxxxxxxxxTrigger is enabled for Client 13 local suppress xx1xxxxxxxxxxxxxTrigger is enabled for Client 14 local suppress x1xxxxxxxxxxxxxxTrigger is enabled for Client 15 local suppress 1xxxxxxxxxxxxxxxTrigger is enabled for Client 16 local suppress All other optionsReserved

65.4.4.1 Client suppress status register (NAR_CSSR)

Clients can be individually suppressed, leaving the rest of the NAR unaffected. When enabled for local suppress, assertion of one of the suppress triggers causes one or more client receive queues to stop queueing messages. A subsequent pulse of the same trigger causes the client(s) to awaken. Local suppress mode is enabled by setting the appropriate bit in one of the NAR_STCRs. An additional status register, Client Suppress Status (NAR_CSSR), shown in the following figure, can then be read to determine which clients are suppressed, and which are active.

Register index: B

	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
R	LSS																0	0	0	0	0	0	0	0	0	0	0	0	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 1135. Client suppress status register (NAR_CSSR)

The NAR_CSSR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress status, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 1074. NAR_CCSR field descriptions

Field	Description
31–16 LSS	Local Suppress Status. 0000000000000000No client is in local suppress mode xxxxxxxxxxxxxxx1Client 1 is suppressed xxxxxxxxxxxxxx1xClient 2 is suppressed xxxxxxxxxxxxx1xxClient 3 is suppressed xxxxxxxxxxxx1xxxClient 4 is suppressed xxxxxxxxxxx1xxxxClient 5 is suppressed xxxxxxxxxx1xxxxxClient 6 is suppressed xxxxxxxxx1xxxxxxClient 7 is suppressed xxxxxxxx1xxxxxxxClient 8 is suppressed xxxxxxx1xxxxxxxxClient 9 is suppressed xxxxxx1xxxxxxxxxClient 10 is suppressed xxxxx1xxxxxxxxxxClient 11 is suppressed xxxx1xxxxxxxxxxxClient 12 is suppressed xxx1xxxxxxxxxxxxClient 13 is suppressed xx1xxxxxxxxxxxxxClient 14 is suppressed x1xxxxxxxxxxxxxxClient 15 is suppressed 1xxxxxxxxxxxxxxxClient 16 is suppressed All other optionsReserved
15–0	Reserved

65.4.5 Receive queue client disable register (NAR_CDR)

Disabling a client reallocates its receive queue to the next active client, allowing that memory space to be put to use. The Client Disable Register, shown in the following figure, allows the user to individually disable or enable clients as needed. The status bits also inform the user when the client has new data available.

Register index: C

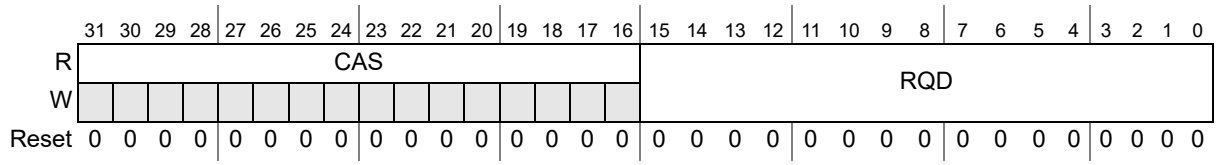


Figure 1136. Client disable register (NAR_CDR)

The NAR_CDR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the client activity status and receive queue disable, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 1075. NAR_CDR field descriptions

Field	Description
31–16 CAS	Client Activity Status. xxxxxxxxxxxxxxx1Client 1 is active with new data xxxxxxxxxxxxxx1xClient 2 is active with new data xxxxxxxxxxxxx1xxClient 3 is active with new data xxxxxxxxxxxx1xxxClient 4 is active with new data xxxxxxxxxxx1xxxxClient 5 is active with new data xxxxxxxxxx1xxxxxClient 6 is active with new data xxxxxxxxx1xxxxxxClient 7 is active with new data xxxxxxxx1xxxxxxxClient 8 is active with new data xxxxxxx1xxxxxxxxClient 9 is active with new data xxxxxx1xxxxxxxxxClient 10 is active with new data xxxxx1xxxxxxxxxxClient 11 is active with new data xxxx1xxxxxxxxxxxClient 12 is active with new data xxx1xxxxxxxxxxxxClient 13 is active with new data xx1xxxxxxxxxxxxxClient 14 is active with new data x1xxxxxxxxxxxxxxClient 15 is active with new data 1xxxxxxxxxxxxxxxClient 16 is active with new data All other optionsReserved
15–0 RQD	Receive Queue Disable. xxxxxxxxxxxxxxx1Disable receive queue for Client 1 xxxxxxxxxxxxxx1xDisable receive queue for Client 2 xxxxxxxxxxxxx1xxDisable receive queue for Client 3 xxxxxxxxxxxx1xxxDisable receive queue for Client 4 xxxxxxxxxxx1xxxxDisable receive queue for Client 5 xxxxxxxxxx1xxxxxDisable receive queue for Client 6 xxxxxxxxx1xxxxxxDisable receive queue for Client 7 xxxxxxxx1xxxxxxxDisable receive queue for Client 8 xxxxxxx1xxxxxxxxDisable receive queue for Client 9 xxxxxx1xxxxxxxxxDisable receive queue for Client 10 xxxxx1xxxxxxxxxxDisable receive queue for Client 11 xxxx1xxxxxxxxxxxDisable receive queue for Client 12 xxx1xxxxxxxxxxxxDisable receive queue for Client 13 xx1xxxxxxxxxxxxxDisable receive queue for Client 14 x1xxxxxxxxxxxxxxDisable receive queue for Client 15 1xxxxxxxxxxxxxxxDisable receive queue for Client 16 All other optionsReserved

65.4.6 Trace Memory fill level and partition configuration register (NAR_AHFPAR)

This register provides the configuration to set the fill level marker for debug trace memory, connected at the trace memory bus, to generate the partial full event indication and the partition related configuration for main queue (internal central queue). The following figure shows the format of the NAR_AHFPAR.

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	AHB01kEN	0	0	0	0	0	0	AHB_01K_Marker			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1137. Trace memory fill level and partition configuration register (NAR_AHFPAR)

The NAR_AHFPAR is described in the following table.

Table 1076. NAR_AHFPAR field description

Field	Description
31-11	Reserved and must be written with 0s
10 AHB01kEN	Enable 1K Marker Event
9-4	Reserved and must be written with 0s
3–0 AHB_01K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 1K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See *Section 65.5.12: Overlay and internal trace memory full status generation* **Note:** This is used in PD as it has memory ranges from 1K to 15K

65.5 Functional description

High-level descriptions of the operation of the NAR and its main functional blocks are given in the following sections.

65.5.1 Reset/Startup

Assertion of global reset causes the NAR operation to immediately cease. All queues and configuration registers are reset. Upon exiting reset, the Nexus Port Controller (NPC) block inside the NAR is disabled. Holding TMS high for five consecutive rising edges of TCK puts TAP controller into the Test-Logic-Reset state regardless of the current state. This applies reset to only TAP controller and JTAG related logic. Applying soft reset through the NAR_CR resets the entire NAR including the TAP controller and configuration registers.

The typical startup procedure for running a trace through the NPC is as follows:

1. Reset the block
2. Upon exiting reset, the Aurora Phy configuration is determined by sampling the configuration pin values from the Nexus Aurora Phy (NAP). If the NAP is enabled, then the NAR is enabled. The Aurora communication is started only when the NAP is ready to accept data. (If any debug lanes are enabled, the Aurora Link begins training at this time.)
3. Debugger/external user sets the NAR enable (NEN) bit in the NAR_CR and configures the NAR. The configuration includes setting up message selection/filtering when two output ports are enabled, enabling timestamp and so on.
4. User programs additional device level debug facilities, such as event-processing, performance-monitoring, and so on.
5. User sets up trace filtering.
6. Enable Nexus transmission in clients.

To enable NAR operation, either the Aurora Link must be enabled at reset, or the user must set the NEN bit in the NAR control register.

Note: The HBDP may already be active by the time the NAR comes up. The NAR only tries to initialize the HBDP if it is currently inactive.

The NAR can be enabled to begin immediate operation, or to start in suppress mode, see *Section 65.5.4, Suppress mode*. Once the NAR and desired ports are enabled, the user should wait until the NAR-ready and HBDP port-ready status bits are set in the NAR status register before beginning Nexus messaging in the clients.

65.5.2 Operation modes

The NAR operating mode is determined by the state of the main control signals: reset and Phy enable. In its enabled state, operation is controlled by its configuration registers. A truth table for all the NAR states is given in this table.

Table 1077. NAR operational modes truth table

Reset	NAR enable	NAR comm disable	Global suppress	HBDP enable	ALT partition	MSP partition	Mode
0	x	x	x	x	x	x	RESET
1	0	x	x	0	x	x	Disabled
1	1	1	x	0	x	x	Trace-Disable
1	x	1	x	1	x	x	Trace-Disable
1	0	x	x	1	x	x	Trace-Disable
1	1	0	1	x	x	x	Suppressed
1	1	0	0	x	Trace Memory bus	x	NAR Transmit
1	1	0	0	1	AUR	x	NAR Transmit
1	1	0	0	1	Trace Memory bus	AUR	NAR Transmit
1	1	0	0	1	AUR	Trace Memory bus	NAR Transmit

There can be overlap between modes as modes are not mutually exclusive.

65.5.2.1 Reset

During reset all internal resources and queues are reset and all output ports are disabled. All inputs are ignored. Upon exiting reset mode, the NAR is in a fully disabled state.

65.5.2.2 Disabled

In the disabled state, all message traffic is ignored, and all output ports are disabled. The NAR is disabled if and only if the NAR enable bit of the NAR control register is not set and the Aurora Link is inactive.

65.5.2.3 Suppressed

The suppressed state differs from the disabled state in that the NAR is on. The NAR pulls out any messages that come up at the clients, keeping the Message Start-End Out (MSEO) bits (so that it can keep track of message boundaries) and discarding the rest.

In the suppressed state, the NAR is waiting for a trigger to begin normal operation. Once that trigger is received, the NAR begins queueing messages that started after the trigger arrived. Incomplete messages that were in transit to the NAR when the trigger arrived are discarded. The NAR can be put into suppressed mode by enabling one of the global suppress states.

65.5.2.4 Trace disable

In trace disable mode, the NAR is powered up but all client-trace is disabled, and no messages are pulled. This mode is also useful for port initialization (to prevent any activity from occurring before the outputs are ready) or for debug/trace-buffer mode, to ensure that, during data retrieval, the NARs queues are stable.

The NAR goes into trace disable (communication-disable) mode when the NAR-enable and NAR communication-disable bits are set or when the HBDP channel is active while the NAR-enable bit is not set.

65.5.2.5 NAR transmit

This is the standard operational mode of the NAR when it is transferring Nexus messages from multiple clients to one or two enabled output port(s). The NAR performs four main operations as described in this table on the incoming messages before transmitting to one of the output ports.

Table 1078. NAR transmit operations

Operation	Description
Data reception	NAR accepts data from each client based on the status of the client receive queue status. The maximum data rate is 8 × 32-bit data words (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word) per cycle per client. MSEO bits of incoming data are monitored to identify end-of-message boundaries.
Input Arbitration	Active receive queue unloads into main queue(s), 8 × 32 bits per cycle (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word). If end-of-message detected, new active receive queue is determined based on client request level or round-robin algorithm.

Table 1078. NAR transmit operations(Continued)

Operation	Description
Output Arbitration	If multiple output ports are enabled, route messages from top of queue to appropriate output port based on message type or source. If split queue is used, outputs can run simultaneously, with each sub-queue communicating directly with a port.
Translation	Data going out the Aurora or trace memory bus ports must be translated into the correct format prior to transmission.

Enabling the NAR and one or more output ports without setting the communication-disable bit puts the NAR into transmit mode.

65.5.3 Data reception and arbitration

The NAR supports 2 clients. Each client has a dedicated input port on the NAR and communicates with its associated input port via a dedicated data bus. A data transmission must consist of an integer number of 32-bit Nexus words (32-bit word is called a beat). The number of beats varies from client to client. The number of beats (Bn) for each client are described in the Calibration and Debug configuration chapter which describes how the modules are configured.

There are two types of clients supported at NAR client interface. The client types have their own protocols to send Nexus messages. The only common parameter is the Nexus word size (beat). One Nexus word consists of 30 Message Data Out (MDO) bits plus two Message Start/End Out (MSEO) control bits. Internally, the NAR converts all client interfaces to a common interface. To achieve this, the NAR instantiates interface level gaskets that handle the protocol conversion and clock domain transfer.

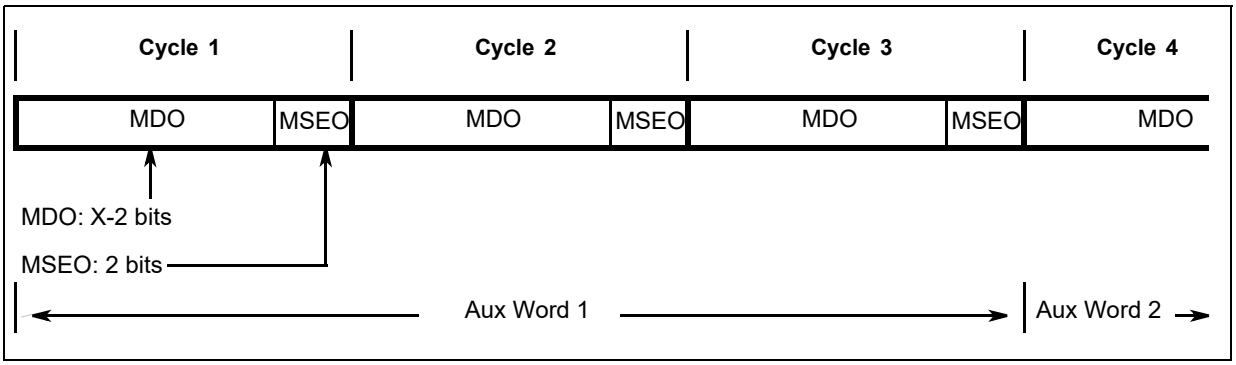
When new data is available in the core, it is transferred over to the asynchronous gasket in the internal NAR clock domain. The gasket then asserts enough data valids to fill internal receive queue, using a receive queue available signal to track available space in the associated queue, under the assumption that the receive queue available signal is initialized with the size of the queue. If the internal arbiter consumes any of its queued entries, it responds with a data accept signal, which increments the receive queue available signal and allows the gasket to send the next piece of data. Under certain circumstances, like receive queue reallocation, it is possible for the internal arbiter to send out surplus data accept signals such that the receive queue available signal can become larger than the size of the queue. If the receive queues are sized correctly, when NAR is actively servicing a particular client, then asserted data accept signals allow the client to continually stream data until the NAR switches to a new client. The data tracking at the gasket assures that it never sends more data than the receive queue can accept and inactive queues are always full and ready for when they are switched to.

The client gaskets must always be in sync with the internal logic of NAR in terms of the number of available entries in each receive queue. For this reason, the internal NPC and its client gaskets need to be reset by the same signal. Any soft-reset of the NAR (via the SRESET configuration bit) is communicated to all gaskets and used to reset every thing inside it.

Each input port has a dedicated receive queue with a minimum of two entries, where an entry is wide enough to store an entire Bn $\times$ 32 data word (where 32 is the number of bits in the Nexus word or the beat). The receive queue can be made larger to allow the NAR to continue transmitting during switches from one client to another in a pipelined environment.

Every cycle, if there is space in a particular receive queue, it latches in one full AUX word from its client. Because the client may be running at a much higher frequency than the NAR, several cycles worth of Nexus messages are contained in a single transaction, as shown in the following figure for a 3-beat per word transaction.

Figure 1138. AUX bus parsing



Each AUX word consists of Bn cycles of Nexus data, along with an optional request-level sideband signal consisting of a priority field (bit 0) and a weight field (bits 1 and 2). Request level priority indicates to the NAR the clients queue status, allowing it to give preferential treatment to clients that are closer to queue overflow. Clients with a set priority bit are always chosen ahead of clients without the bit. For this reason, priority should be used sparingly, only when a client is in danger of overflowing. Request-level weighting also gives a client preferential treatment, but without entirely shutting out the other clients. Normally, the NAR attempts to switch to a new client every time it sees an end-of-message boundary.

As the AUX word is pulled into the receive queue, the MSEO bits are monitored to determine the current Nexus state of the data flow; since Nexus is a variable-length protocol, messages can end in any cycle. If the start message or end message state is detected on any data beat, the beat is marked as containing an end-of-message boundary.

Any received beat which is in the idle state, indicating null data content, is discarded (see the IEEE-ISTO 5001-2003 standard for details of the Nexus protocol). At the same time, if two outputs are enabled, then the source and TCODE fields are extracted from each new message and compared with the user-programmed filters. Messages that generate a match are tagged with an MSP bit, which allows the NAR to route it to the correct port.

Every cycle, the NAR transfers 8 beats of data from the receive queues into the main output queue. Only one port (the active port) is allowed to empty into the main queue at a time: the active port unloads its receive queue into the main queue until the end of the current Nexus message is detected. At that point, a new active port is selected based on the latest request level received at each port. In the case where multiple ports have the same request level (or when request level is not used), then the active port is rotated amongst those ports. If the end of message occurs in the middle of an AUX word, then all the beats up to the end of message are transferred from the old active port, while the rest of the beats come from the new active port. For example, if the message ended at beat two of a three-beat AUX word, then the first two beats into the main queue would come from the original active port, while the last beat would come from the new active port.

65.5.4 Suppress mode

The suppress mode feature enables the user to quickly turn Nexus messaging through the NAR on or off, based on the four dedicated suppress-mode triggers. Suppress mode is only

supported in the PD NAR, it is not supported in the BD NAR. A suppress can either be global, meaning messaging is turned off for all clients, or local, which only applies to selected clients. In suppress mode, the NAR pulls messages from the suppressed client(s), but does not queue or deliver them. After extracting the MSEO bits to keep track of message boundaries, the rest of the message is discarded. This allows the NAR to better keep up with fast-running Nexus clients without having to turn off messaging in the client.

Typically, a client would be suppressed during less interesting periods, and would be awakened when some preset condition is detected. This is communicated to the NAR via the one of the suppress triggers, which can be sourced by a client or an external Event Processing block. When an enter suppress trigger is received, the NAR output queue finishes servicing the current active ports until the end of the message. If the end of the message occurs in a middle beat of a Nexus word, then the remainder of the word is padded with null messages. Once the last message has been loaded into the output queue, the receive queues are all emptied. New messages are pulled from clients as they appear, go through their respective receive queues and then are discarded. In this way, the NAR can naturally monitor the MSEO bits and keep track of each client's Nexus state. In suppress mode, the output ports remain active until the output queue is emptied, meaning there is a delay between the assertion of the suppress trigger and the end of messaging.

When the NAR is in suppress mode and receives a wake-up trigger, it continues discarding any partial messages still in the receive queues. Any message that begins after the trigger is queued normally, insuring that only complete messages are sent to the output queue.

Global suppress mode applies to all clients. Any of the suppress triggers can be programmed to cause the NAR to toggle between suppressed and awake by setting the GST bit of the associated Suppress Trigger Control Register (NAR_STCR). Additionally, the NAR starts up in global suppress mode if the PUS bit of the Control Register is set when the NAR is enabled. All clients are then suppressed until a wake-up trigger is received. If the PUS bit is set without enabling any of the triggers for global suppress, a configuration error results and the NAR powers up in active mode.

Clients can go also into local suppress mode without putting the entire NAR to sleep. Each NAR_STCR has a 16-bit LST field. Setting one or more of these bits causes the trigger assigned to that NAR_STCR to toggle the associated clients between suppressed and awake. All other clients are unaffected. Thus, suppress trigger one can be assigned to one set of clients, while suppress trigger two can be assigned to another set of clients, and different parts of the device can be ignored at different times, all based on different, programmable events. Global suppress always has higher priority than local suppress. That is, if the NAR is in global-suppressed mode, any local suppress wake-up triggers are ignored. Also, if some clients are in local suppress when a global suppress trigger hits, then all go into suppress mode; however, upon the subsequent global wake-up, the clients that were previously in local suppress remain so.

65.5.5 Stall detection

Since the NAR can only switch between clients on an end-of-message boundary, every message that goes into the NAR must complete. If a client happens to die or stall in the middle of transmitting a message, this uncompleted message could cause the NAR to stall, at a time when other clients might still be able to deliver useful information. To prevent this from occurring, the NAR monitors each client for a stall condition, which is defined as occurring when data is not valid in the middle of a Nexus message for a set number of cycles.

When a stall condition is detected, the NAR can do one thing. If the client stalls, then the NAR must complete the current stalled message to continue operating. This scenario is termed a stall error. On a stall error, the NAR injects a dummy message beat into the message stream after the last valid beat of data received from the stalled client. The dummy message is a single beat injected at the output of the receive queue into the main queue. The MSEO bits of the data payload are set to 2b11 regardless of the previous Nexus state. This forces the dummy message into the End Message state. When the NAR injects a dummy message, it also generates an error message to notify the user what has happened.

Injecting a dummy message is an intrusive act, which alters the data stream from the client. If a stalled client is thought likely to recover cleanly, it might be advisable to suppress stall errors to prevent the data stream from being corrupted. This can be done by setting the SED bit of the configuration register, which carries the risk of a truly dead client causing the NAR to hang. Otherwise, the default operation is for the NAR to inject the dummy message and then monitor the client for signs of recovery. If the client does come back online, the NAR attempts to wait until it sees an end-of-message boundary before capturing new data. At best, if the client recovers smoothly, only the interrupted message is lost; if the client does not recover smoothly, it might not be possible for the client and NAR to get back in sync in terms of their Nexus states, which would lead to nothing but garbled data coming out. For this reason, it is also possible to prevent the NAR from trying to recover from a stall error, by setting the SRD bit of the NAR_CR. If the SRD bit is set, a stalled client is not allowed to recover until the SRD bit is unset or the NAR is reset.

65.5.6 Output arbitration

The NAR allows two output ports to be active at the same time. This feature is only available on the BD NAR. Typically, this feature is only useful if one of the ports is used to service messages of a specific type and/or messages from a particular client, while the other port supports all the rest of the message traffic. For example, the NAR could be set up to send in-circuit trace (ICT) traffic out the HBDP and to reserve the trace memory bus port for all other Nexus messages.

If two output ports are enabled, one of them must be designated as the Message Select Port (MSP). The MSP is the destination for those messages that satisfy a set of user-defined requirements as to the source and type of message. All other traffic is sent to the other enabled port. Ports are enabled by setting the port enable bits in the BD NAR_CR. The MSP is designated by setting bits [10–9] and ALT is designated by setting bits [7–6]. Once the MSP has been specified, other bits in the NAR_CR, along with the Source Filter Register (NAR_SFR) and Type Filter Register (NAR_TFR), are used to determine which messages go to the MSP and which do not. These facilities can be used to build a logical expression that can filter messages based on up to four types and/or four sources.

The NAR_SFR can be programmed with up to four source codes, while the NAR_TFR can hold up to four message-type codes. Each filter also includes an enable bit to turn it on or off. A source match occurs if a messages source code matches one of the source filters or (if the SFI bit is set) if it does not match any of the source filters. Similarly, a type match occurs if a messages type code matches one of the type filters or (if the TFI bit is set) if it does not match any of the type filters. Finally, the ACC bit of the NAR_SFR determines when both source and type filters are used, whether the two are ANDed together (ACC = 0)

or ORed together (ACC = 1). For example, to send ICT messages out the HBDP and all other traffic to the trace memory bus port:

- Set ALT = 10 in the NAR_CR—sets trace memory bus port as the alternate port
- Set MSP = 01 in the NAR_CR—sets HBDP as the message select port
- Set FE0 = 1 & TF0 = 100010 (0x22) in the NAR_TFR—enables type filter 0 to match ICT messages

To send everything but data trace from source 13 to the trace memory bus port, while all other traffic goes to the HBDP:

- Set ALT = 01 in the NAR_CR—sets HBDP as the alternate port
- Set MSP = 10 in the NAR_CR—sets trace memory bus port as the message select port
- Set TFI = 1—inverts type-filter logic
- Set FE0 = 1 & TF0 = 000101 (0x05) in the NAR_TFR to select data trace of size 8 bits
- Set FE1 = 1 & TF1 = 000110 (0x06) in the NAR_TFR to select data trace of size 16 bits
- Set FE2 = 1 & TF2 = 001101 (0x0d) in the NAR_TFR to select data trace of size 32 bits
- Set FE3 = 1 & TF3 = 001110 (0x0e) in the NAR_TFR to select data trace size of 64 bits
- Set FE0 = 1 & SF0 = 001101 (0x0d) in the NAR_SFR to select source trace ID = 13
- Set ACC = 0 in NAR_SFR so the final filter is ~(data_trace) AND (source = 13)HBDP

Data is written into the output queue at a maximum rate of one entry per cycle. If the current messages source and type pass the filters, then the message is marked for the MSP. When the entry is later pulled from the queue, it is then sent to the MSP. Unmarked entries are sent to the alternate port. Message destination must be recalculated on every end-of-message boundary, using the fixed-width SRC and TYPE fields at the beginning of each Nexus message.

65.5.7 Message translation and formatting

The NAR performs the formatting on Nexus messages before sending them to an output port. The exact translation performed depends on the destination port.

65.5.7.1 Aurora format

Nexus messages are pre-translated before sending to the HBDP using the Aurora protocol. The NAR output is a stream of 16 × 4-bit words (where 4 is the width of the NAR Aurora out port) that are further encoded by an external Aurora block before being sent for transmission on the Aurora (HBDP) port.

Entries from the output queue are chopped into chunks of 4 × 16-bit words. The amount of data that can be transmitted per Aurora cycle depends on the number of lanes available in the HBDP (transmitted HBDP word = 16 × number of lanes).

65.5.7.2 Trace memory bus format

The trace memory bus works on AHB-EW protocol. Data destined for the trace memory bus port is split back out into 32-bit beats (Nexus word). Depending on the relative sizes of the beat and the width of the trace memory bus port, one or more Nexus words are packed into a trace memory bus data word, with any remaining bits padded to 0.

For example, if the size of the beat = 32 and the width of the trace memory bus = 64, then each trace memory bus data packet contains two beats of Nexus data. Since the data is stored in a regular format (MSEO bits in same place at every location), this makes data

interpretation straightforward, regardless of where the trace memory bus port is sending the data. The trace memory bus translation block performs the necessary accumulation and resizing in the likely case where the trace memory bus width does not equal the queue entry size.

65.5.8 Aurora interface

The external Nexus Aurora Link (NAL) contains all of the link-layer, PCS, and control logic for the HBDP channel. The Aurora data is transmitted from the NAR to the NAL for sending to the HBDP. Typically, the NAL powers up coming out of reset, but it can also be enabled by the NAR by turning on the NAR with the HBDP enabled. The reverse is true as well. If the NAL is active while the NAR is disabled, then the NAR powers up to communication-disabled mode.

Data flow between the NAR and the NAL uses a simple, synchronous interface. If the NAL is active, it indicates this by a ready signal from the NAL. If Aurora is configured as output it is indicated by a signal from the NAR.

The Aurora interface can either be operated in framing or streaming mode. Streaming mode provides the highest performance and lowest communication overhead through the link. Whereas framing mode provides greater robustness and improved error-handling at the receive side of the link. When framing mode is enabled, the NAR encapsulates the Aurora data in frames consisting of 256 Aurora chunks; that is, after every 256 Aurora transmissions, the NAR asserts end of frame and start of frame to mark the end of the last frame and the beginning of the next frame. In either mode, the NAR always asserts a start-of-frame when beginning transmission, and an end-of-frame when its output queue empties, to notify the channel partner of the end of valid data.

65.5.9 Trace memory bus output port usage

The NAR can communicate directly to external memory by sending Nexus data out onto the trace memory bus port, using the AHB-EW (Early Write) protocol. This provides a convenient, secondary area for data in applications or in situations where the HBDP is not available. The trace memory bus access is generated internally by trace memory bus controller.

Because in many systems it is desirable to have the trace memory bus interface run at a divided clock rate relative to the main platform clock, the trace memory bus port operates off of a separate AHB clock. This clock is assumed to be free-running (i.e. unaffected by NAR-enable and the status of the Aurora Link), synchronous to the platform clock used by the rest of the NAR, and either running at the same rate as the platform clock or an integer divide of that rate.

The trace memory bus interface protocol is a slightly modified version of AMBA AHB specification. The write data comes in the address phase instead of data phase and it is termed as Early Write (EW). When NAR uses the trace memory bus port as a destination for Nexus data, it performs a series of writes to a predefined block of memory space.

This memory space is specified through the trace memory bus base address and trace memory bus max transfer configuration registers. The size of the block is set by the max transaction count (MXFR field in NAR_TCR) times the size of a transaction (trace memory

bus-width, 64-byte, 128-byte, 256-byte, or 384-byte, depending on block transfer mode and block transfer size). For example, if:

- Base address = 0x0D00_0000 (i.e. TBAHI = 0x0, TBALO = 0x0D00_0000)
- NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
- NAR_TCR[BTS] = 00 (Block-transfer size = 64-byte)
- NAR_TCR[MXFR] = 0x100 (max transaction count)

then the total block size would be 0x100 $\times$ 0x40 = 0x4000 bytes. The dedicated debug address range would then be 0x0D00_0000 – 0x0D00_3FFF.

The NAR can either write to the debug space one (trace memory bus-width) word at a time, or for improved performance, it can use a block transfer. If the BTM bit of the trace memory bus Control Register (NAR_TCR) is set, then the NAR uses block transactions to send data through the trace memory bus port, where the size of the block is determined by the setting of the BTS field of the NAR_TCR.

In block transfer mode, the NAR waits until the main queue partition dedicated to the trace memory bus port reaches a specified threshold point, determined by the BTH field of the NAR_TCR, before initiating a block transfer. The default threshold is the size of the block transfer. That is, a block transfer does not start until there is enough data in the associated partition to fill the block. But it might be desirable, for bandwidth reasons or because the associated partition is smaller than the block transfer size, to begin the transfer sooner. Hence, based on BTH, a transfer can begin when the partition is at 25%, 50%, or 75% of the block transfer size. This capability should be used with caution. If the Nexus client does not generate enough new data during the transfer to fill the block, then a block transfer error results.

The NAR generates Nexus trace write transactions for the trace memory bus port using the address from the base address register for the first transaction and then incrementing through the memory space. Once the programmed max number of transactions have been sent, the NAR sets the trace memory bus overflow status bit and, if desired, asserts the NAR event-out. At this point, the NAR can either shut down the trace memory bus port for trace, or, if the TBW bit of the NAR_TCR is set, reset the address to the beginning of the debug space and continue sending trace data. This allows the debugger to view the most recent data, in cases where the buffer overflows.

When TBW bit of NAR_TCR is not set then NAR shuts down the trace memory bus port. To restart the trace memory write again from trace memory base address, the user should write either the NPC trace memory bus base address (TBALO/TBAHI) or the transaction count (MXFR). The user does not have to actually change the value, but can write the same target base address again. The NAR sees any write to one of these registers as a reconfiguration of the trace memory bus port, and resets its internal address pointer to the base address along with clearing the target-space-full flag.

65.5.10 Internal message generation

Generally, the NAR simply conveys Nexus messages from client to port, and does not generate messages of its own. There are, however, certain situations when the NAR needs to communicate directly with the debugger. Internally generated messages are added to the queue the same as external messages. If the queue is partitioned, then internal messages are always channeled to whichever partition is dedicated to the HBDP.

The NAR generates the following message types:

- Device_ID—The first message transmitted when the NAR comes up. It consists of a 6-bit TCODE (TCODE = 1) and the 32-bit device ID. The first beat of the message is always an IDLE (MSEO = 2b11, MDO = all 0s) to allow the debugger Nexus state machine to correctly initialize (by moving to end message or idle state). Device_ID is also generated if the queue is repartitioned (by changing either the ALT or MSP setting in the NAR_CR) while it is not empty. This serves the dual purpose of insuring that the user always gets at least one device_ID message and providing a message boundary in the case where there might have been corruption due to the lost data.
- Watchpoint—The NAR generates its own watchpoint match message under the following circumstances as shown in *Table 1082*.
- NAR_Error (TCODE = 8)—If an error condition is detected inside the NAR, it generates its own error message. An NAR_Error message consists of the 6-bit TCODE, the 6-bit NAR source ID, and a 32-bit data field which corresponds to the NAR status register. An NAR error can be triggered by:
  - HBDP error
  - Illegal configuration
  - Stall Error
  - Trace memory bus read/write access error

This table describes the NAR message formats.

Table 1079. NAR message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Device ID	6	6	fixed	TCODE	TCODE value = 1
	32	32	fixed	DID	DID register value
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message(1)	6	6	fixed	TCODE	TCODE value = 8
	6	6	fixed	SRC(2)	NAR source ID
	4	4	fixed	ETYPE	See *Table 1080*
	12	12	fixed	ECODE	See *Table 1081*
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Watchpoint Hit Message(3)	6	6	fixed	TCODE	Value = 15
	6	6	fixed	SRC(2)	NAR source ID
	6	6	fixed	WPHIT	See *Table 1082*
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Error message conforms to the IEEE-ISTO 5001-2011 standard.
2. Tools should treat the NAR SRC field as a 4-bit SRC field and a 2-bit unused field. The extra 2-bit field (the two most significant bits of the 6-bit TCODE) is always 0b00. This operation is not compliant to the IEEE-ISTO 5001 Nexus standard.
3. The Watchpoint Hit message does not conform to the IEEE-ISTO 5001-2011 standard, it conforms to the IEEE-ISTO 5001-2003 standard. The WPHIT should be a variable length packet, but is implemented as a fixed length packet. In most cases, tools will not see a difference, however, a difference will be seen if a timestamp field is appended to the message.

65.5.10.1 Error conditions

NAR errors can occur upon events such as illegal memory accesses, block-transfer overloads, communication errors, or when the NAR is configured incorrectly. If the configuration registers are programmed incorrectly, it is possible to put the NAR into an illegal state. The NARs response varies depending on the error. For a minor error, the NAR reverts to a default mode and continues operation, while major errors result in complete shutdown.

In either case, a Nexus error message is sent to the Aurora out port, if its enabled, otherwise to the trace memory bus port. The Nexus error message contains both ETYPE and ECODE fields:

- ETYPE is used to denote the basic type of error that has occurred
- ECODE carries specific additional information about the error

All NAR error types are defined in this table.

Table 1080. NAR error types

ETYPE	Description	Classification	NAR action	Comment
0000 (h'0)	Receive queue overrun	Minor	Dropped message	If a client sends a data word that the NAR cannot accept (due to a full receive queue), the new data word is dropped.
0001 (h'1)	Internal message contention	Minor	Dropped message	Internal errors/watchpoints occurred faster than NAR could generate messages for them, causing lower-priority message(s) to be dropped. The type of message dropped is conveyed in the ECODE field, see *Table 1081*.
0011 (h'3)	Trace memory bus port read/write access error	Minor	Access fails	Unable to send an Aurora-to-trace memory bus request. Typical causes would be writes larger than the data buffer or sending in a new request before the previous one completes.(1)
0101 (h'5)	Invalid NRR	Minor	Access fails	Tried to access an un-implemented NRR.
1001 (h'9)	Config error	Major	See *Table 1081*	The specific configuration error is conveyed in bits 0–3 of the ECODE field (*Table 1081*), as well as the NCE field of Status Register.
1100 (h'C)	HBDP error	Minor	N/A	A hard error is automatically reset the Aurora Link and causes it to re-initialize. Although the Link is down, the error message is queued and released as soon as the Link comes back up.
1110 (h'E)	Block transfer error	Minor	Dummy fill	If the Trace memory bus port becomes data-starved during a block transfer, it fills the rest of the block with IDLE messages.
1111 (h'F)	Stall error	Minor	—	—

1. In the case where a request was successfully translated to a Trace memory bus transaction which itself returned an error, that is conveyed in the STATUS field of the Nexus Target Response message subsequently returned to the debugger.

Typically, the ECODE field is used to communicate which type of message was dropped in queue overflow or message contention situation, but for the NAR it is also used to denote the type of configuration error that occurred. If a configuration error is detected, the NAR generates a Config Error Nexus message and transmits it, with the actual type of error encoded in the four MSBs of ECODE, as well as the NCE field of the Status Register. The ECODE field encodings are listed in this table.

Table 1081. Configuration error codes

ECODE	ETYPE	Description	NAR action	Comment
0000xxxxxxx1	0001	Watchpoint message dropped	N/A	Occurs with internal message overflow error.
0000xxxxxx1x	0001	Data trace message dropped	N/A	Occurs with internal message overflow error; MMA response message was dropped.
00001xxxxxxx	0001	Response message dropped	N/A	Occurs with internal message overflow error; target response (memory access or NRR) message was dropped.
0000xxx1xxxx	0001	Status message dropped	N/A	Occurs with internal message overflow error.
000100000000	1001	Config error: multiply defined port	Communication disable mode	Both MSP and ALT were assigned to the same port. NPC waits until the configuration is fixed before proceeding.
001000000000	1001	Config error: no ALT defined	Tracebuffer	An MSP was defined without defining the ALT. The NPC reverts to tracebuffer mode.
001100000000	1001	Config error: cannot enable port	Communication disable mode	User has attempted to enable a port that does not exist in this implementation of the NPC.
010000000000	1001	Config error: cannot support message filtering	Message filtering and/or queue partitioning disabled	User has attempted to enable message filtering or queue partitioning with only one output enabled. Filtering/partitioning is turned off.
100100000000	1001	Config error: no wakeup trigger	Power up in active mode	Occurs if none of the defined clients in the system can cause a wakeup (PUS is set in NAR_CR, but no NAR_STCRs have WUT set).

65.5.10.2 Watchpoint messages

The NAR generates its own watchpoint match message under the circumstances shown in the following table. The value of the Watchpoint Hit field of the Watchpoint Message is also shown in this table.

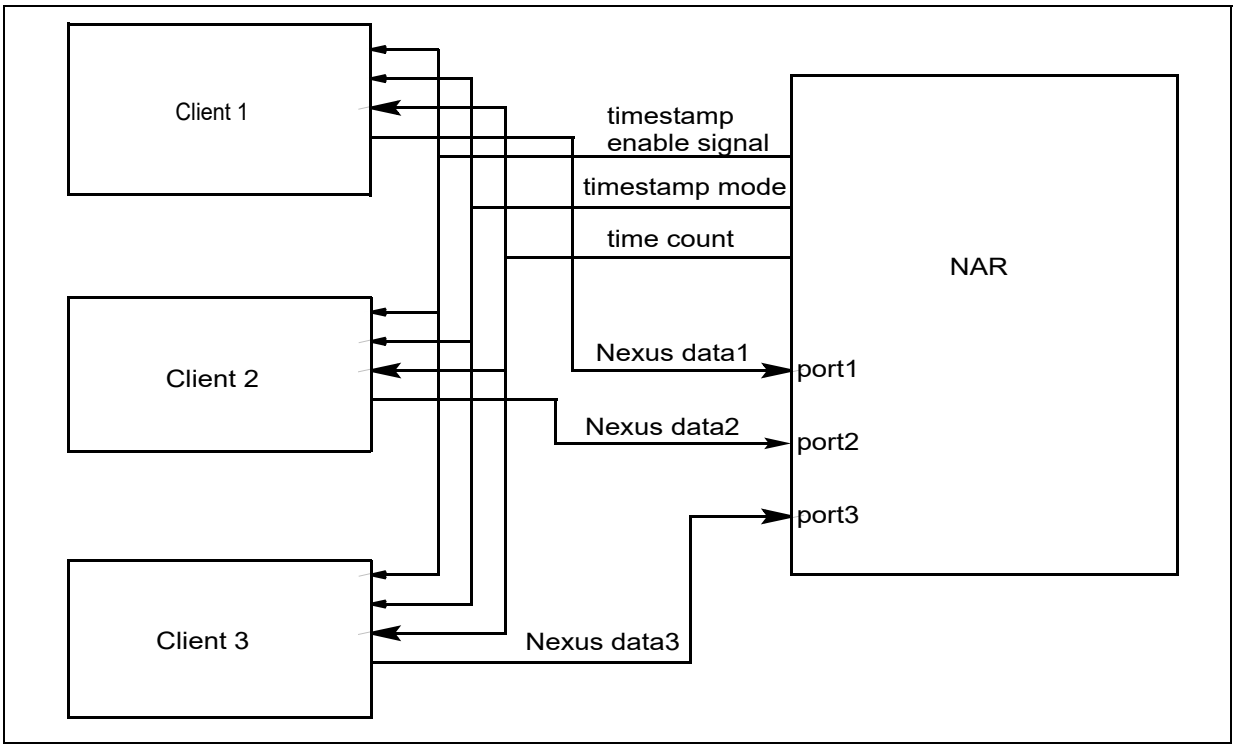
Table 1082. Watchpoint Hit values

WPHIT	Description	Comment
b010000	Entering Suppress Mode	If (local or global) suppress mode is enabled and the appropriate trigger is received, the NAR generates a watchpoint message. The type of event (in this case, a suppress event) is encoded in the WPHIT field of the watchpoint message (see the Nexus standard, sec. 5.3.21).
b100000	Client Sync	If sync messaging is enabled, the NAR generates a sync watchpoint message whenever a client sync event occurs.
b000100	Trace Memory bus Block Overflow (TBO)	If the trace memory bus block is filled, causing it to either disable trace memory bus data trace or wrap back to the beginning of the dedicated debug memory block and if trace memory bus overflow messaging is enabled, then an TBO message is generated.
b001000	Tracebuffer Wrap	If the NAR is being used in tracebuffer mode with the wrap-around feature, then a watchpoint message is generated every time the tracebuffer wraps back to address 0. This would be useful in informing the user that the buffer had wrapped and that older data had been overwritten.

65.5.11 Timestamp function

The NAR and all of its clients are required to transmit timestamps, to allow precise ordering of events. All clients synchronous or asynchronous uses NAR timer value broadcasted over a bus, this ensures that all timestamps are in sync as shown in *Figure 1139*. The NAR uses enable signal to centrally control the enabling of time stamping, which informs the clients to start adding the timestamp field to their message traffic.

Figure 1139. Timestamp architecture



Timestamping is enabled by setting the TEN bit of the NAR_CR. This causes the NAR to assert the time stamp enable signal to clients. Typically, the user would also set the TRS bit at this time to reset the NAR Time stamp counter. Internally the NAR has a 30-bit counter external to the NPC block. This 30-bit counter is broadcast to all clients after encoding its value in Gray code. This minimizes the error in synchronization by the clients. The NPC uses an additional internal 24-bit counter to send the timestamp inside its messages. The client converts the gray coded timestamp to binary to append to the Nexus messages.

The Time stamp mode, a 2-bit field, is broadcasted to all client which tells the granularity of sending time stamps into message. This is controlled by the TSG bits of the NAR_CR.

Asynchronous clients, those on a separate clock domain from the NAR, synchronize the enable, mode and timestamp value in their domain. The error introduced by this is assumed to be within tolerable range by application.

A timestamp counter reset can be forced by causing a 0 to 1 transition in the TRS bits of the NAR_CR. The NAR deasserts the enable signal to clients while the reset occurs, and then the counter is restarted (assuming the TEN bit is set). Data throughput can be increased (at the loss of timestamp granularity) by controlling the TSG bits of the NAR_CR. Rather than adding a timestamp to every message, clients can be told to only stamp every 4th, 16th, or 32nd message, easing the bandwidth requirements on the output ports.

The timestamp block maintains the timestamp counter for NAR. It is the central time source for all synchronous/asynchronous clients.

65.5.12 Overlay and internal trace memory full status generation

In certain events, such as main memory (internal trace memory) full or external overlay trace memory full, it is required to control the clients to stop sending data, to avoid any loss of messages from the clients.

The NAR provides three outputs (triggers) to indicate three different situations as described below:

- Internal trace memory (the main queue) is full
- External debug memory on trace memory bus is full
- External debug memory on trace memory bus is full up to a programmable watermark set by the configuration register

The last two triggers are used by SPU to generate stall and suppress triggers. Typically the stall request should be generated whenever space in the external memory connected to trace memory bus has been exhausted. The NAR starts writing from the first location (wrap) as specified in trace memory bus address register in this condition if overwrite trace memory bus debug memory (TBW bit) is set in the NAR_TCR.

The user should configure the partial full watermark register such that it can give around 90% full indication of external debug memory on the trace memory bus. The emulator tool memory full indication is typically generated from the DCI block which actually communicates with the tool.

For example, the following steps can be used to generate event at around 90% fill level for trace memory connected in AHB-EW and to set the watermark in the configuration register.

- From the configuration, the programmer knows the following:
  - AHB start (base) address—Base address = 0x0D00_0000 (i.e.TBAHI = 0x0, TBALO = 0x0D00_0000
  - Block transfer size—NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
  - Max transfer count—NAR_TCR[MXFR] = 0xC0 (maximum transaction count)
- Thus, the total block size is (0xC0 $\times$ 0x40) = 0x3000 bytes. The dedicated debug address range is then 0x0D00_0000 – 0x0D00_2FFF.
- To generate an event at nearly 90% fill level of the trace memory at 1 KB boundary, the value to be programmed in the register NAR_AHFPAR[3:0] is calculated as:
  - Number of 1K blocks in 0x3000 bytes = (3000 right shift by 10 bits) = C = 12
  - So the value to be programmed should be 11 as 90% of 12 is almost 11.
- NAR_AHFPAR[3:0] = (NAR_TBALO[13:10] + 0xB) = (0x7+0xB) 0x2 (upper bits truncated due to width) and enable this with NAR_AHFPAR[10] = 1.

This assumes that NAR starts writing from base address and stops at maximum address and no reading is done in overlay memory trace data area during this operation. Whenever the address bus bits [13:10] match with this value, the partial event is generated (toggled).

65.5.13 Configuration/debug interface JTAG (1149.1)

The configuration and status registers of the NAR are accessed through the standard IEEE 1149.1 JTAG interface. The JTAG frequency (TCK) must be 1/2 of the NAR operating clock. The access mechanism through JTAG is described in this section.

The NAR block uses the IEEE 1149.1-2001 TAP for accessing registers. TAP signals include TCK, TDI, TMS, and TDO. The NAR implements a TAP controller state machine that transitions based on the state of the IEEE 1149.1-2001 16-state state machine. It also implements Nexus controller state machine as defined by the IEEE-ISTO 5001-2001 standard.

The instructions implemented by the NAR TAP controller are listed in *Table 1083*. Each unimplemented instruction acts like the BYPASS instruction. The size of the NAR instruction register is 4 bits.

Figure 1140. Nexus controller state machine

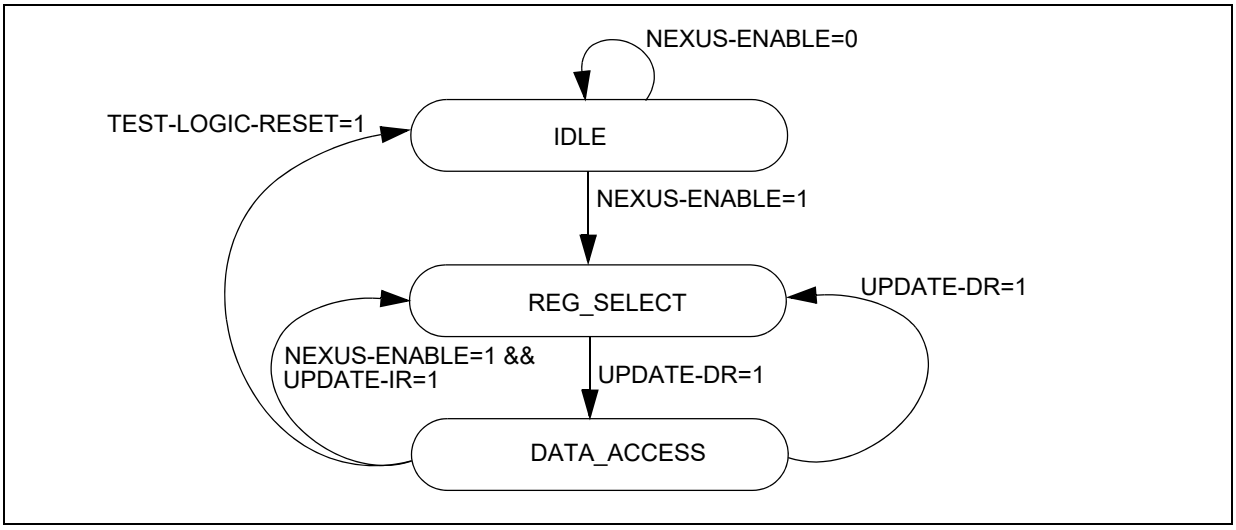
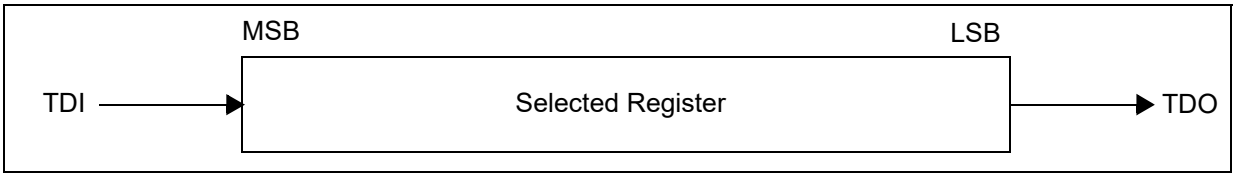


Table 1083. Implemented instructions

Instruction Name	Private/ Public	Opcode	Description
NEXUS_ENABLE	Public	0x0	Activate Nexus controller state machine to read and write NAR registers.
BYPASS	Private	0x0F	NAR BYPASS instruction. Also the value loaded into the NAR IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

Figure 1141. Shifting data into register



65.5.13.1 Enabling the NAR TAP controller

Assertion of the power-on reset signal resets the NAR TAP controller. When not in power-on reset the NAR TAP controller is enabled by driving JCOMP with the NAR enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.

65.5.13.2 Loading NEXUS-ENABLE instruction

Access to the NAR 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 1140*, transitions to the REG_SELECT state. The Nexus

controller has three states: idle, register select, and data access. The following table illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Table 1084. 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.
5 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

65.5.13.3 Selecting a JTAG register to create a bus (memory or peripheral) access

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows the development tool access to all NAR 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 1142*. The read/write control bit is set to 1 for writes and 0 for reads. The index for registers are shown in *Table 1065*.

Figure 1142. Nexus command format for register selection

MSB	LSB
7-bit register index	R/W

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.

This table illustrates a sequence which writes a 32-bit value to a register.

Table 1085. Writing a 32-bit value to a 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 4 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

65.5.14 Trace memory bus controller

The purpose of the trace memory bus is to transfer trace data from main queue to external overlay memory. This port works on a modified AHB protocol that supports early write. It has AHB interface signals as defined in AMBA specification (Rev 2.0). In addition, there are side-band signals for added functionality.

Key features of the module include:

- AHB-EW Interface
  - Early Write (for higher BW in special module)
  - Configurable 32/64-bit address bus width
  - 64-bit data bus width
  - Transfer Types
  - IDLE, NONSEQ, SEQ
  - Single Transfers
  - Burst Transfers
  - Configurable support for INCR
  - Configurable support for INCR4, INCR8, INCR16, WRAP4
  - Response Type
  - OKAY, ERROR

  - Transfer Size
  - Byte, Half Word, Word, Double-Word
  - Protection Control
  - Cacheable/ Non Cacheable Access Control
  - User/Privilege Access
- Endianess Support
  - Configurable Little Endian/Big Endian support on AMBA-AHB interface
- Pipeline Support
  - Total of two transactions (One plus one ongoing)

65.5.14.1 Operation

The trace memory bus controller allows all read/write operations to be cut through. The controller has a pipeline depth of two. While the data phase of one transaction is ongoing, it can accept one more read/write request from the internal bus master and initiate the address phase on the AHB interface during the last data cycle of the ongoing transaction.

65.5.14.2 Write operation

When the trace memory bus initiator asserts a request for a write operation, it can find the acknowledge signal parked asserted if the controller is ready to accept the transaction. If the pipeline is full or in case of an ongoing read transaction, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB interface once it accepts the transaction by asserting the acknowledge on the trace memory bus interface. The write data from the trace memory bus initiator is accepted by the gasket by asserting the acknowledge only on completion of address phase at AHB interface, which is signified by receiving the ready signal asserted. Simultaneously, the data is sent to the data bus. Once the requisite number of bytes has been accepted from the trace memory bus initiator, the gasket generates end of data and transfer according to the Trace memory bus protocol.

The AHB protocol supports data transactions of byte, half-word, word and double-word (when both trace memory bus and AHB data bus width is 64 bits). A write transaction from the trace memory bus initiator take place as single/multiple transactions on the AHB side.

65.5.14.3 Read operation

When the trace memory bus initiator asserts a request signal to request for a read operation, it can find the acknowledge signal parked asserted if the gasket is ready to accept the transaction. If the pipeline is full, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB-EW interface once it accepts the transaction by asserting the acknowledge on trace memory bus interface. The controller reads the data from the bus when the ready signal is sensed asserted. The data is transferred on the trace memory bus data bus with the data acknowledge asserted. The gasket asserts end of data and transfer at the end of transaction as per trace memory bus protocol.

The AHB protocol supports data transactions of byte, word, half-word and double-word (when both trace memory bus and AHB data bus width is 64 bits). A read transaction from trace memory bus initiator takes place as single/multiple transaction on the AHB interface before the read data is merged and sent on the trace memory bus read data bus.

In case of variable data bus width on trace memory bus and AHB interface (32-bit data bus on one interface and 64-bit data bus on the other interface), a maximum of 64 bits is read on each beat.

65.5.15 Legacy client interface controller

This block converts the legacy client protocol into internal client protocol that is accepted by the internal NPC block. The Nexus beat size must be same for both interfaces. The Nexus beat is composed of MDO bits along with two MSEO bits. The trace data width from the legacy client is fixed at one beat. The architecture of this block is shown in *Figure 1143*.

The legacy client sends each message after arbitrating the bus. It requests the gasket with a 2-bit request signal whenever it has data to send. A 00 means no request, 01 a low-priority request, 10 a medium-priority request, and 11 a high-priority request. The legacy client can send one maximum message sized data on receiving a grant from the block. There is no backpressure mechanism during transfer of a message from the client to this block. The block asserts a grant if it has at least one max message size space in its FIFO. Seeing the grant the client asserts busy signal and holds it until it wants to send data, it may hold it up to max message size data is transferred.

The following architecture describes the legacy client to internal client protocol conversion block. There is a small FIFO of parameterizable depth used to store the data from client. The block asserts a grant only when at least one max message sized space is available in FIFO. The write happens in the client's clock domain. The write pointer is passed to other domain which is NPC clock domain.

The logic on other side compares read and write pointer to check the data availability in the block's FIFO and space availability in NPC's receive queue. If space and data both are available it tries to send message to the NPC.

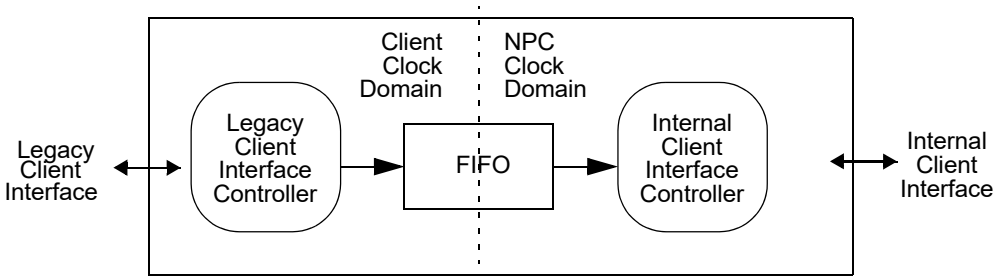


Figure 1143. Legacy client Interface controller

The size of the receive queue of the corresponding client at NPC must be known to the block. The block drives a not empty signal whenever it has data in FIFO and looks for a ready signal. It can send as many Nexus beats as receive queue can store. It operates on decrements a counter after sending data to the NPC so that it can track how much data it can send. The counter is incremented by the value of acknowledge from NPC which tells how may spaces are freed in receive queue.

65.6 Initialization information

The NAR is initialized by asserting the block reset signal for a minimum of one cycle of the slowest clock. Reset clears all latches to 0, with the only exception being the output queue, whose initialization state depends on the reset behavior of the compiled memory used. Upon exiting reset, the NAR is in the disabled state, as described in *Section 65.5.2.1, Reset*.

Pointers and state machines are initialized by the starting of NAR functional clocks (which occurs either upon the setting of the NAR enable bit or upon the assertion of the Phy enable signal) or by an NAR soft reset.

65.7 Application information

The base application for the NAR is collecting and collating Nexus trace data. Programming examples for how to set up the NAR for trace are given in *Section 65.5.6, Output arbitration*.

66 e200z215An3 Nexus 3 Module

The e200z215An3 Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port that is used in conjunction with the JTAG port in these processors.

66.1 Introduction

66.1.1 General Description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

66.1.2 Terms and Definitions

Table 1086 contains a set of terms and definitions associated with the Nexus 3 module.

Table 1086. Terms and Definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.

Table 1086. Terms and Definitions(Continued)

Term	Description
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	“System-on-a-Chip”. SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase “according to the standard” is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that 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.

66.1.3 Feature List

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

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated.

An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.

- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- External hardware trigger inputs to independently enable/disable Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace Messaging
- Auxiliary interface for higher data input/output
  - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
  - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
  - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
  - One (1) Read/Write Error pin (**nex_err_b**) pin
  - One (1) Watchpoint Event output pin (**nex_evto_b**)
  - Four (4) additional Watchpoint Event output pins (**nex_wevto[3:0]**) for SoC use
  - One (1) Event In pin (**nex_evti_b**)
  - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)

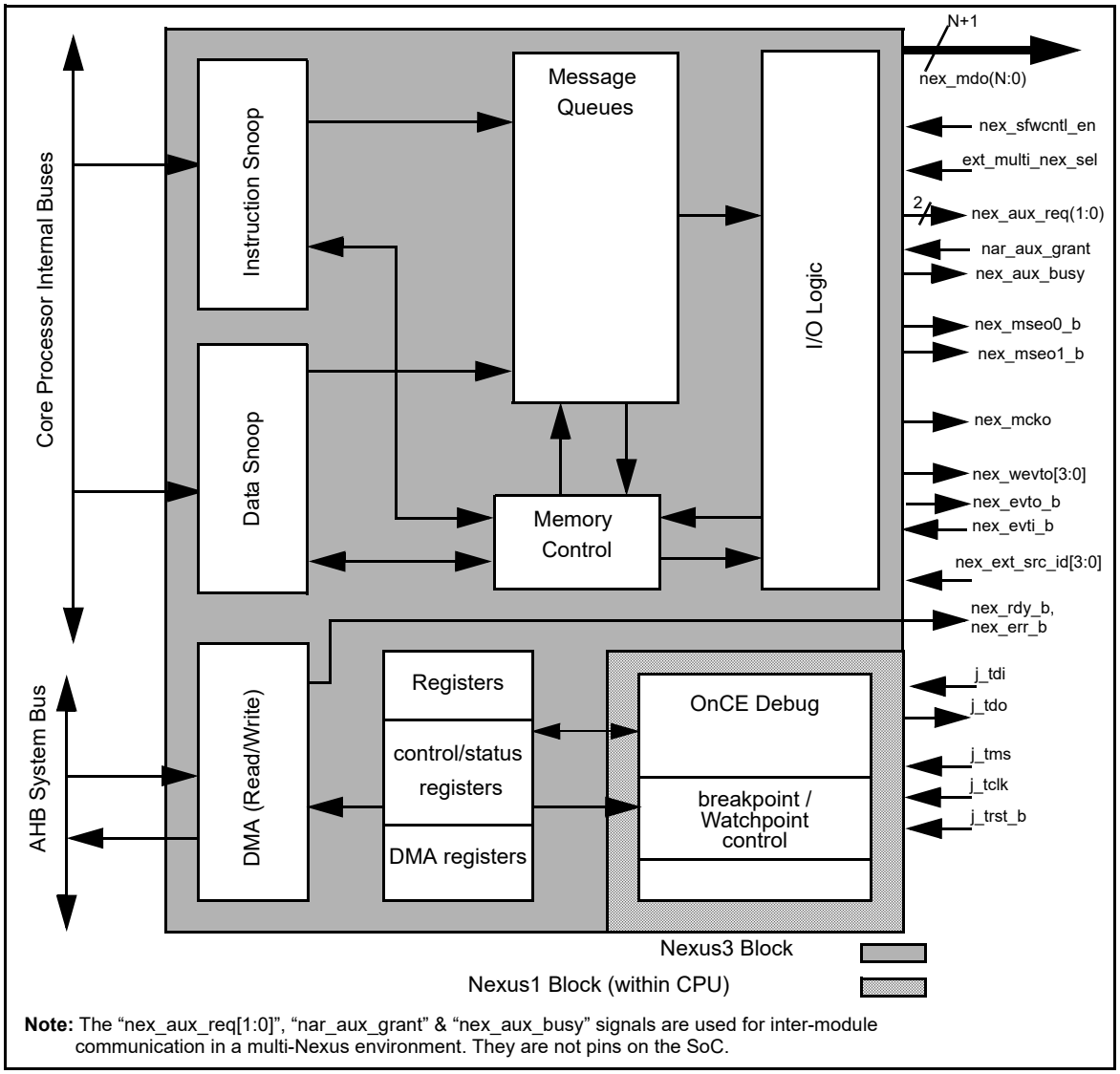
Note: For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (@ the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM — maximum number of MDO pins) or Reduced Port Mode (RPM — minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note: 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.

66.1.4 Functional Block Diagram

Figure 1144. Nexus 3 functional block diagram



Note: The “nex_aux_req[1:0]”, “nar_aux_grant” & “nex_aux_busy” signals are used for inter-module communication in a multi-Nexus environment. They are not pins on the SoC.

66.2 Enabling Nexus 3 Operation

The Nexus module is enabled by loading a single instruction (NEXUS3-ACCESS) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the **nex_evti_b** input signal is asserted at the time that **j_trst_b** is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

66.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

66.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in *Table 1087*.

Table 1087. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
Program Trace — Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
Program Trace — Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
Data Trace — Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 1092*)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace — Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 1092*)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register

Table 1087. Supported TCODEs(Continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
Program Trace — Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
Program Trace — Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Program Trace — Indirect Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Data Trace — Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 1092*)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)

Table 1087. Supported TCODEs(Continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Trace — Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 1092*)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to *Table 1090*) — indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See Section *Section 66.12.4: Resource Full Messages*.)
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 66.12.1: Branch Trace Messaging types*)
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 66.12.1: Branch Trace Messaging types*.)

Table 1087. Supported TCODEs(Continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
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 1091*)
	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 66.12.5: Program Correlation Messages*.)
	0	32	CDATA2	variable	correlation data field 2— PID info (See *Section 66.12.5: Program Correlation Messages*.)

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 66.12.1: Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 1088 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 1088. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 1089 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 1089. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 – 1111	Reserved

Table 1090 shows the encodings used for resource codes for certain messages.

Table 1090. 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 1091 shows the event code encodings used for certain messages.

Table 1091. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010–0011	Reserved for future functionality
0100	Disabling Program Trace
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110–1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM} = 0 and $DC4_{PTMARK}$= 1
1010	Branch and link occurrence (direct branch function call)
1011–1111	Reserved for future functionality

Table 1092 shows the data trace size encodings used for certain messages.

Table 1092. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 — no data
0001	Byte

Table 1092. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)(Continued)

DTM Size Encoding	Transfer Size
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001–1111	Reserved

66.4 Nexus 3 Programmer's Model

This section describes the Nexus 3 programmers model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See *Section 66.5: Nexus 3 Register Access via JTAG/OnCE* and *Section 66.6: Nexus 3 Register Access via Software*for details on Nexus 3 register access.

Note: Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

Table 1093 details the register map for the Nexus 3 module.

Table 1093. Nexus 3 Register Map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #(1)
Client Select Control (CSC)(2)	0x1	R	0x02	—	
Port Configuration Register (PCR)(3)	PCR_INDEX(2)	R/W	—	—	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	—
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	—
Reserved	0x8	R/W	0x18	0x19	—
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	—
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	—
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	—
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377

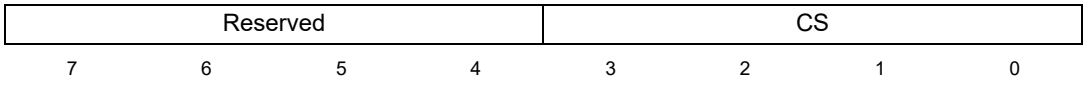
Table 1093. Nexus 3 Register Map(Continued)

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #(1)
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 → 0x2F	—	0x28→0x5E	0x29→0x5F	—
Development Status (DS)	0x30	R	0x60	—	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	—	0x68	0x69	—
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 → 0x3F	—	0x72→0x7E	0x73→0x7F	—

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.
3. The "PCR_INDEX" is a parameter determined by the SoC.

66.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.



Nexus Reg# 0x1; Read-only; Reset: 0x0

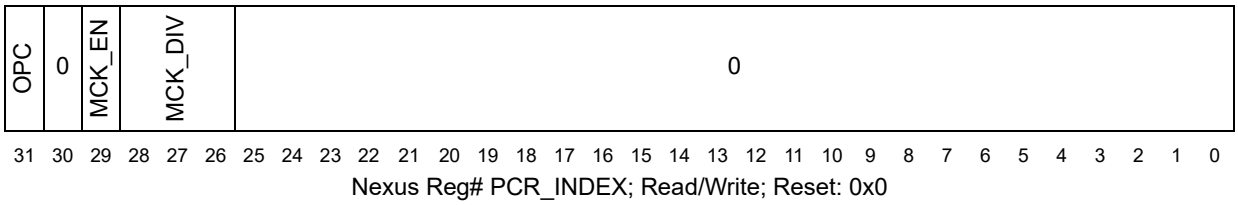
Figure 1145. Client Select Control (CSC) register

Table 1094. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0*xx* Nexus client (SoC level)

66.4.2 Port Configuration Register (PCR) — reference only

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.



Nexus Reg# PCR_INDEX; Read/Write; Reset: 0x0

Figure 1146. Port Configuration Register (PCR)

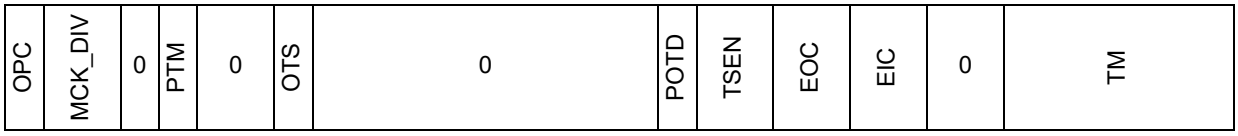
Table 1095. PCR field descriptions

Bit	Name	Description
31	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)
30	—	Reserved for future functionality
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled
28:26	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.
25:0	—	Reserved for future functionality

Note: The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

66.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in *Figure 1147* and its fields are described in *Table 1096*.



Nexus Reg# 0x2; DCR: 368; Read/Write; Reset: 0x0

Figure 1147. Development Control 1 (DC1) register

Table 1096. DC1 field descriptions

Bits	Name	Description
31	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
30:29	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.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 — Program Trace uses traditional Branch Messages 1 — Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 — PID0 data is transmitted within Ownership Trace Messages 1 — Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality
14	POTD	Periodic Ownership Trace Disable 0 — Periodic Ownership Trace message events are enabled 1 — Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable — (not implemented, write to 00) 00 — Timestamp is disabled
11:10	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 1*x* — Reserved
9:8	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 1*x* — Reserved

Table 1096. DC1 field descriptions(Continued)

Bits	Name	Description
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode(1) 000000 — All Trace Disabled *xxxxx*1 — Ownership Trace enabled *xxxx*1*x* — Data Trace enabled *xxx*1*xx* — Program Trace enabled *xx*1*xxx* — Watchpoint Trace enabled *x*1*xxxx* — Reserved 1*xxxxx* — 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 66.4.7: Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers* for more information on watchpoint triggering. Refer to *Section 66.16: External Hardware Trigger Controls* 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.

66.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z215An3) Core Debug Support chapter.

Development Control Register 2 is shown in *Figure 1148* and its fields are described in *Table 1097*.

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	EWC

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

Nexus Reg# 0x3; DCR: 369; Read/Write; Reset: 0x0

Figure 1148. Development Control 2 (DC2) register

Table 1097. DC2 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[3]** 0001 — Watchpoint #0 triggers **nex_wevto[3]** 0010 — Watchpoint #1 triggers **nex_wevto[3]** 0011 — Watchpoint #2 triggers **nex_wevto[3]** 0100 — Watchpoint #3 triggers **nex_wevto[3]** 0101 — Watchpoint #4 triggers **nex_wevto[3]** 0110 — Watchpoint #5 triggers **nex_wevto[3]** 0111 — Watchpoint #6 triggers **nex_wevto[3]** 1000 — Watchpoint #7 triggers **nex_wevto[3]** 1001 — Watchpoint #8 triggers **nex_wevto[3]** 1010 — Watchpoint #9 triggers **nex_wevto[3]** 1011 — Watchpoint #10 triggers **nex_wevto[3]** 1100 — Watchpoint #11 triggers **nex_wevto[3]** 1101 — Watchpoint #12 triggers **nex_wevto[3]** 1110 — Watchpoint #13 triggers **nex_wevto[3]** 1111 — Watchpoint #14 triggers **nex_wevto[3]**
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[2]** 0001 — Watchpoint #0 triggers **nex_wevto[2]** 0010 — Watchpoint #1 triggers **nex_wevto[2]** 0011 — Watchpoint #2 triggers **nex_wevto[2]** 0100 — Watchpoint #3 triggers **nex_wevto[2]** 0101 — Watchpoint #4 triggers **nex_wevto[2]** 0110 — Watchpoint #5 triggers **nex_wevto[2]** 0111 — Watchpoint #6 triggers **nex_wevto[2]** 1000 — Watchpoint #7 triggers **nex_wevto[2]** 1001 — Watchpoint #8 triggers **nex_wevto[2]** 1010 — Watchpoint #9 triggers **nex_wevto[2]** 1011 — Watchpoint #10 triggers **nex_wevto[2]** 1100 — Watchpoint #11 triggers **nex_wevto[2]** 1101 — Watchpoint #12 triggers **nex_wevto[2]** 1110 — Watchpoint #13 triggers **nex_wevto[2]** 1111 — Watchpoint #14 triggers **nex_wevto[2]**
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[1]** 0001 — Watchpoint #0 triggers **nex_wevto[1]** 0010 — Watchpoint #1 triggers **nex_wevto[1]** 0011 — Watchpoint #2 triggers **nex_wevto[1]** 0100 — Watchpoint #3 triggers **nex_wevto[1]** 0101 — Watchpoint #4 triggers **nex_wevto[1]** 0110 — Watchpoint #5 triggers **nex_wevto[1]** 0111 — Watchpoint #6 triggers **nex_wevto[1]** 1000 — Watchpoint #7 triggers **nex_wevto[1]** 1001 — Watchpoint #8 triggers **nex_wevto[1]** 1010 — Watchpoint #9 triggers **nex_wevto[1]** 1011 — Watchpoint #10 triggers **nex_wevto[1]** 1100 — Watchpoint #11 triggers **nex_wevto[1]** 1101 — Watchpoint #12 triggers **nex_wevto[1]** 1110 — Watchpoint #13 triggers **nex_wevto[1]** 1111 — Watchpoint #14 triggers **nex_wevto[1]**

Table 1097. DC2 field descriptions(Continued)

Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[0]** 0001 — Watchpoint #0 triggers **nex_wevto[0]** 0010 — Watchpoint #1 triggers **nex_wevto[0]** 0011 — Watchpoint #2 triggers **nex_wevto[0]** 0100 — Watchpoint #3 triggers **nex_wevto[0]** 0101 — Watchpoint #4 triggers **nex_wevto[0]** 0110 — Watchpoint #5 triggers **nex_wevto[0]** 0111 — Watchpoint #6 triggers **nex_wevto[0]** 1000 — Watchpoint #7 triggers **nex_wevto[0]** 1001 — Watchpoint #8 triggers **nex_wevto[0]** 1010 — Watchpoint #9 triggers **nex_wevto[0]** 1011 — Watchpoint #10 triggers **nex_wevto[0]** 1100 — Watchpoint #11 triggers **nex_wevto[0]** 1101 — Watchpoint #12 triggers **nex_wevto[0]** 1110 — Watchpoint #13 triggers **nex_wevto[0]** 1111 — Watchpoint #14 triggers **nex_wevto[0]**
15:0	EWC	EVTO Watchpoint Configuration(1) 0000000000000000 No Watchpoints #0–15 trigger **nex_evto_b** *xxxxxxxxxxxxxxx*1 Watchpoint #0 triggers **nex_evto_b** *xxxxxxxxxxxxxx*1*x* Watchpoint #1 triggers **nex_evto_b** *xxxxxxxxxxxxx*1*xx* Watchpoint #2 triggers **nex_evto_b** *xxxxxxxxxxxx*1*xxx* Watchpoint #3 triggers **nex_evto_b** *xxxxxxxxxxx*1*xxxx* Watchpoint #4 triggers **nex_evto_b** *xxxxxxxxxx*1*xxxxx* Watchpoint #5 triggers **nex_evto_b** *xxxxxxxxx*1*xxxxxx* Watchpoint #6 triggers **nex_evto_b** *xxxxxxxx*1*xxxxxxx* Watchpoint #7 triggers **nex_evto_b** *xxxxxxx*1*xxxxxxxx* Watchpoint #8 triggers **nex_evto_b** *xxxxxx*1*xxxxxxxxx* Watchpoint #9 triggers **nex_evto_b** *xxxxx*1*xxxxxxxxxx* Watchpoint #10 triggers **nex_evto_b** *xxxx*1*xxxxxxxxxxx* Watchpoint #11 triggers **nex_evto_b** *xxx*1*xxxxxxxxxxxx* Watchpoint #12 triggers **nex_evto_b** *xx*1*xxxxxxxxxxxxx* Watchpoint #13 triggers **nex_evto_b** *x*1*xxxxxxxxxxxxxx* Watchpoint #14 triggers **nex_evto_b** 1*xxxxxxxxxxxxxxx* 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 1149* and its fields are described in *Table 1098*.

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	0	EWC

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

Nexus Reg# 0x4; DCR: 370; Read/Write; Reset: 0x0

Figure 1149. Development Control 3 (DC3) register

Table 1098. DC3 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #15–#26 trigger **nex_wevto[3]** 0001 Watchpoint #15 triggers **nex_wevto[3]** 0010 Watchpoint #16 triggers **nex_wevto[3]** 0011 Watchpoint #17 triggers **nex_wevto[3]** 0100 Watchpoint #18 triggers **nex_wevto[3]** 0101 Watchpoint #19 triggers **nex_wevto[3]** 0110 Watchpoint #20 triggers **nex_wevto[3]** 0111 Watchpoint #21 triggers **nex_wevto[3]** 1000 Watchpoint #22 triggers **nex_wevto[3]** 1001 Watchpoint #23 triggers **nex_wevto[3]** 1010 Watchpoint #24 triggers **nex_wevto[3]** 1011 Watchpoint #25 triggers **nex_wevto[3]** 1100 Watchpoint #26 triggers **nex_wevto[3]** 11*x*1 Reserved
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 No Watchpoints #15–#26 trigger **nex_wevto[2]** 0001 Watchpoint #15 triggers **nex_wevto[2]** 0010 Watchpoint #16 triggers **nex_wevto[2]** 0011 Watchpoint #17 triggers **nex_wevto[2]** 0100 Watchpoint #18 triggers **nex_wevto[2]** 0101 Watchpoint #19 triggers **nex_wevto[2]** 0110 Watchpoint #20 triggers **nex_wevto[2]** 0111 Watchpoint #21 triggers **nex_wevto[2]** 1000 Watchpoint #22 triggers **nex_wevto[2]** 1001 Watchpoint #23 triggers **nex_wevto[2]** 1010 Watchpoint #24 triggers **nex_wevto[2]** 1011 Watchpoint #25 triggers **nex_wevto[2]** 1100 Watchpoint #26 triggers **nex_wevto[2]** 11*x*1 Reserved
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 No Watchpoints #15–#29 trigger **nex_wevto[1]** 0001 Watchpoint #15 triggers **nex_wevto[1]** 0010 Watchpoint #16 triggers **nex_wevto[1]** 0011 Watchpoint #17 triggers **nex_wevto[1]** 0100 Watchpoint #18 triggers **nex_wevto[1]** 0101 Watchpoint #19 triggers **nex_wevto[1]** 0110 Watchpoint #20 triggers **nex_wevto[1]** 0111 Watchpoint #21 triggers **nex_wevto[1]** 1000 Watchpoint #22 triggers **nex_wevto[1]** 1001 Watchpoint #23 triggers **nex_wevto[1]** 1010 Watchpoint #24 triggers **nex_wevto[1]** 1011 Watchpoint #25 triggers **nex_wevto[1]** 1100 Watchpoint #26 triggers **nex_wevto[1]** 1101 Watchpoint #27 triggers **nex_wevto[1]** 1110 Watchpoint #28 triggers **nex_wevto[1]** 1111 Watchpoint #29 triggers **nex_wevto[1]**

Table 1098. DC3 field descriptions(Continued)

Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 No Watchpoints #15–#29 trigger **nex_wevto[0]** 0001 Watchpoint #15 triggers **nex_wevto[0]** 0010 Watchpoint #16 triggers **nex_wevto[0]** 0011 Watchpoint #17 triggers **nex_wevto[0]** 0100 Watchpoint #18 triggers **nex_wevto[0]** 0101 Watchpoint #19 triggers **nex_wevto[0]** 0110 Watchpoint #20 triggers **nex_wevto[0]** 0111 Watchpoint #21 triggers **nex_wevto[0]** 1000 Watchpoint #22 triggers **nex_wevto[0]** 1001 Watchpoint #23 triggers **nex_wevto[0]** 1010 Watchpoint #24 triggers **nex_wevto[0]** 1011 Watchpoint #25 triggers **nex_wevto[0]** 1100 Watchpoint #26 triggers **nex_wevto[0]** 1101 Watchpoint #27 triggers **nex_wevto[0]** 1110 Watchpoint #28 triggers **nex_wevto[0]** 1111 Watchpoint #29 triggers **nex_wevto[0]**
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration(1) 00000000000000 No Watchpoints #16–#29 trigger **nex_evto_b** *xxxxxxxxxxxxx*1 Watchpoint #16 triggers **nex_evto_b** *xxxxxxxxxxxx*1*x* Watchpoint #17 triggers **nex_evto_b** *xxxxxxxxxxx*1*xx* Watchpoint #18 triggers **nex_evto_b** *xxxxxxxxxx*1*xxx* Watchpoint #19 triggers **nex_evto_b** *xxxxxxxxx*1*xxxx* Watchpoint #20 triggers **nex_evto_b** *xxxxxxxx*1*xxxxx* Watchpoint #21 triggers **nex_evto_b** *xxxxxxx*1*xxxxxx* Watchpoint #22 triggers **nex_evto_b** *xxxxxx*1*xxxxxxx* Watchpoint #23 triggers **nex_evto_b** *xxxxx*1*xxxxxxxx* Watchpoint #24 triggers **nex_evto_b** *xxxx*1*xxxxxxxxx* Watchpoint #25 triggers **nex_evto_b** *xxx*1*xxxxxxxxxx* Watchpoint #26 triggers **nex_evto_b** *xx*1*xxxxxxxxxxx* Watchpoint #27 triggers **nex_evto_b** *x*1*xxxxxxxxxxxx* Watchpoint #28 triggers **nex_evto_b** 1*xxxxxxxxxxxxx* Watchpoint #29 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

66.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in *Figure 1150* and its fields are described in *Table 1099*.

PTMARK	DTMARK	0	EVCDM

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

Nexus Reg# 0x5; DCR: 371; Read/Write; Reset: 0x0

Figure 1150. Development Control 4 (DC4) register

Table 1099. DC4 field descriptions

Bits	Name	Description
31	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'
30	DTMARK	Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask(1) 0000000000000000 No EVCODEs masked for Program Correlation Messages *xxxxxxxxxxxxxxx*1 EVCODE #0 is masked for Program Correlation Messages *xxxxxxxxxxxxxx*1*x* EVCODE #1 is masked for Program Correlation Messages *xxxxxxxxxxxxx*1*xx* EVCODE #2 is masked for Program Correlation Messages *xxxxxxxxxxxx*1*xxx* EVCODE #3 is masked for Program Correlation Messages *xxxxxxxxxxx*1*xxxx* EVCODE #4 is masked for Program Correlation Messages *xxxxxxxxxx*1*xxxxx* EVCODE #5 is masked for Program Correlation Messages *xxxxxxxxx*1*xxxxxx* EVCODE #6 is masked for Program Correlation Messages *xxxxxxxx*1*xxxxxxx* EVCODE #7 is masked for Program Correlation Messages *xxxxxxx*1*xxxxxxxx* EVCODE #8 is masked for Program Correlation Messages *xxxxxx*1*xxxxxxxxx* EVCODE #9 is masked for Program Correlation Messages *xxxxx*1*xxxxxxxxxx* EVCODE #10 is masked for Program Correlation Messages *xxxx*1*xxxxxxxxxxx* EVCODE #11 is masked for Program Correlation Messages *xxx*1*xxxxxxxxxxxx* EVCODE #12 is masked for Program Correlation Messages *xx*1*xxxxxxxxxxxxx* EVCODE #13 is masked for Program Correlation Messages *x*1*xxxxxxxxxxxxxx* EVCODE #14 is masked for Program Correlation Messages 1*xxxxxxxxxxxxxxx* EVCODE #15 is masked for Program Correlation Messages

1. Refer to *Table 1091* for implemented EVCODEs.

66.4.6 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.

Nexus Reg# 0x4; DCR: 409; Read-only; Reset: 0x0

Figure 1151. Development Status (DS) register

Table 1100. DS field descriptions

Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting State (**p_waiting** signal asserted 010 Halted State (**p_halted** signal asserted 011 Reserved 1*xx* Reserved
27:26	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)
25:0	—	Reserved for future functionality (read as 0)

66.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allow the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related “window” for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4’b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For

example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z215An3) Core Debug Support chapter.

PTS	PTE	DTS	DTE	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

Nexus Reg# 0xB; DCR: 375; Read/Write; Reset: 0x0

Figure 1152. Watchpoint Trigger (WT) register

Table 1101 details the Watchpoint Trigger register fields.

Table 1101. WT field descriptions

Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
27:24	PTE	Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.

PTST

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

Nexus Reg# 0x35; DCR: 412; Read/Write; Reset: 0x0

Figure 1153. Program Trace Start Trigger Control (PTSTC) register

Table 1102 details the PTSTC register fields.

Table 1102. PTSTC field descriptions

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #27 *xxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #28 *xx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #29 *x*1*xxxxxxxxxxxxxxxx* Use Watchpoint #30 1*xxxxxxxxxxxxxxxx* Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

PTET

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

Nexus Reg# 0x36; DCR: 413; Read/Write; Reset: 0x0

Figure 1154. Program Trace End Trigger Control (PTETC) register

Table 1103 details the PTETC register fields.

Table 1103. PTETC field descriptions

Bits	Name	Description
31:0	PTET	Program Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #27 *xxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #28 *xx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #29 *x*1*xxxxxxxxxxxxxxxx* Use Watchpoint #30 1*xxxxxxxxxxxxxxxx* Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

0	DTST

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

Nexus Reg# 0x37; DCR: 414; Read/Write; Reset: 0x0

Figure 1155. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 1104. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #27 *xxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #28 *xx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

0	DTET

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

Nexus Reg# 038; DCR: 415; Read/Write; Reset: 0x0

Figure 1156. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 1105. DTETC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #27 *xxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #28 *xx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #29 *x*1*xxxxxxxxxxxxxxxx* Use Watchpoint #29 1*xxxxxxxxxxxxxxxx* Use Watchpoint #29

66.4.8 Nexus Watchpoint Mask (WMSK) register

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).

WEM

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

Nexus Reg# 0x33; DCR: 411; Read/Write; Reset: 0x0

Figure 1157. Watchpoint Mask (WMSK) register

Table 1106 details the Watchpoint Trigger register fields.

Table 1106. WMSK field descriptions

Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging 00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Watchpoint #0 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Watchpoint #1 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Watchpoint #2 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Watchpoint #3 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Watchpoint #4 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Watchpoint #5 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Watchpoint #6 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Watchpoint #7 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Watchpoint #8 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Watchpoint #9 enabled for WTM *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Watchpoint #10 enabled for WTM *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Watchpoint #11 enabled for WTM *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Watchpoint #12 enabled for WTM *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Watchpoint #13 enabled for WTM *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Watchpoint #14 enabled for WTM *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Watchpoint #15 enabled for WTM *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Watchpoint #16 enabled for WTM *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxx* Watchpoint #17 enabled for WTM *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxx* Watchpoint #18 enabled for WTM *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxx* Watchpoint #19 enabled for WTM *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxx* Watchpoint #20 enabled for WTM *xxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxx* Watchpoint #21 enabled for WTM *xxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxx* Watchpoint #22 enabled for WTM *xxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #23 enabled for WTM *xxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #24 enabled for WTM *xxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #25 enabled for WTM *xxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #26 enabled for WTM *xxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #27 enabled for WTM *xxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #28 enabled for WTM *xx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #29 enabled for WTM *x*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #30 enabled for WTM 1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #31 enabled for WTM

66.4.9 Nexus Overrun Control (OVCR) register

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.

0	SPTHOLD	0	SPEN	0	STTHOLD	0	STEN
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

Nexus Reg# 0x32; DCR: 410; Read/Write; Reset: 0x0

Figure 1158. Nexus Overrun Control (OVCR) register

Figure 1159. OVCR field descriptions

Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	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
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 Suppression is disabled *xxxxx*1 Ownership Trace message suppression is enabled *xxxx*1*x* Data Trace message suppression is enabled *xxx*1*xx* Program Trace message suppression is enabled *xx*1*xxx* Watchpoint Trace message suppression is enabled *x*1*xxxx* Reserved 1*xxxxx* Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	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
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 Stalling is disabled 1 Stalling is enabled

66.4.10 Data Trace Control (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.

RWT1	RWT2	RWT3	RWT4	0	STDC1	STDC2	STDC3	STDC4	0	RC1	RC2	RC3	RC4	DI	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

Nexus Reg# 0xD; DCR: 376; Read/Write; Reset: 0x0

Figure 1160. Data Trace Control (DTC) register

The following table details the Data Trace Control register fields.

Table 1107. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1 1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range 1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range

Table 1107. DTC field descriptions(Continued)

Bits	Name	Description
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses **Note:** The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. **Note:** The setting of the DI bit also affects the information provided on the data trace port outputs (See the “Data Trace Port Signals” section in the Core (e200z215An3) Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

66.4.11 Data Trace Start Address Registers (DTSA1–4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

Data Trace Start Address																															
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

Nexus Reg# 0xE; DCR: 377; Read/Write; Reset: 0x0

Figure 1161. Data Trace Start Address 1 (DTSA1) register

Data Trace Start Address																															
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

Nexus Reg# 0xF; DCR: 378; Read/Write; Reset: 0x0

Figure 1162. Data Trace Start Address 2 (DTSA2) register

Data Trace Start Address																															
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

Nexus Reg# 0x10; DCR: 379; Read/Write; Reset: 0x0

Figure 1163. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address																															
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

Nexus Reg# 0x11; DCR: 380; Read/Write; Reset: 0x0

Figure 1164. Data Trace Start Address 4 (DTSA4) register

66.4.12 Data Trace End Address 1–4 (DTEA1–4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Data Trace End Address																															
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

Nexus Reg# 0x12; DCR: 381; Read/Write; Reset: 0x0

Figure 1165. Data Trace End Address 1 register

Data Trace End Address																															
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

Nexus Reg# 0x13; DCR: 382; Read/Write; Reset: 0x0

Figure 1166. Data Trace End Address 2 register

Data Trace End Address																															
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

Nexus Reg# 0x14; DCR: 383; Read/Write; Reset: 0x0

Figure 1167. Data Trace End Address 3 register

Data Trace End Address																															
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

Nexus Reg# 0x15; DCR: 408; Read/Write; Reset: 0x0

Figure 1168. Data Trace End Address 4 register

Table 1108 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 1108. Data trace — address range options

Programmed values	Range Control Bit Value	Range Selected
DTSA < DTEA	0	DTSA → ← DTEA
DTSA < DTEA	1	← DTSA DTEA →
DTSA > DTEA	N/A	Invalid Range — no trace
DTSA = DTEA	N/A	Invalid Range — no trace

Note: DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are inclusive of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are exclusive of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI} = 1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (i.e. when the instruction prefetch is discarded).

66.4.13 Read/Write Access Control/Status (RWCS) register

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. Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values. The RWCS Register also provides Read/Write Access Status information per *Table 1110*.

AC	RW	SZ	MAP	PR	ATTR	0	CNT	ERR	DV
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

Nexus Reg# 0x7; Read/Write(1); Reset: 0x0

Figure 1169. Read/Write Access Control/Status (RWCS) register

1. ERR and DV are read-only

Table 1109. RWCS field descriptions

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 1*xx* Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map All other values are reserved
RWCS[23:22]	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
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	Reserved for future functionality

Table 1109. RWCS field descriptions(Continued)

Bits	Name	Description
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR (2)	Read/Write Access Error (see *Table 1110*)
RWCS[0]	DV(2)	Read/Write Access Data Valid (see *Table 1110*)

1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 1110. 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

66.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

Read/Write Data

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

Nexus Reg# 0xA; Read/Write; Reset: 0x0

Figure 1170. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. *Table 1111* shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 1111. RWD data placement for transfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			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

Table 1111. RWD data placement for transfers(Continued)

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Doubleword first RWD pass (low order data) second RWD pass (high order data)	0 0 0	0 1 1	X X	X X	X X	X X

Table Notes:
"X" indicates byte lanes with valid data
"—" indicates byte lanes that will contain unused data.

Table 1112 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 1112. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	—	—	—	AHB[7:0]
Byte @001	0 0 1	—	—	—	AHB[15:8]
Byte @010	0 1 0	—	—	—	AHB[23:16]
Byte @011	0 1 1	—	—	—	AHB[31:24]
Byte @100	1 0 0	—	—	—	AHB[39:32]
Byte @101	1 0 1	—	—	—	AHB[47:40]
Byte @110	1 1 0	—	—	—	AHB[55:48]
Byte @111	1 1 1	—	—	—	AHB[63:56]
Half @000	0 0 0	—	—	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	—	—	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	—	—	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	—	—	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000 first RWD pass second RWD pass	0 0 0	 AHB[31:24] AHB[63:56]	 AHB[23:16] AHB[55:48]	 AHB[15:8] AHB[47:40]	 AHB[7:0] AHB[39:32]

Table Notes:
"—" indicates byte lanes that will contain unused data.

66.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

Read/Write Address

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

Nexus Reg# 0x9; Read/Write; Reset: 0x0

Figure 1171. Read/Write Access Address (RWA) register

66.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in *Table 1093*.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See *Section 66.23: IEEE 1149.1 (JTAG) RD/WR sequences*.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see *Table 1093*), 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 1172. JTAG DR format

(7 bits)	(1 bit)
Nexus Register Index	R/W

RESET Value: 0x00

Table 1113. JTAG DR field descriptions

Nexus Register Index	Selected from values in *Table 1093*
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.

66.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these accesses only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. *Table 1093* shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

66.7 Nexus Message Fields

Nexus messages are comprised of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

66.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

66.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

66.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant ‘1’. To recreate the original address, the relative address is XORed with the previously decoded address.

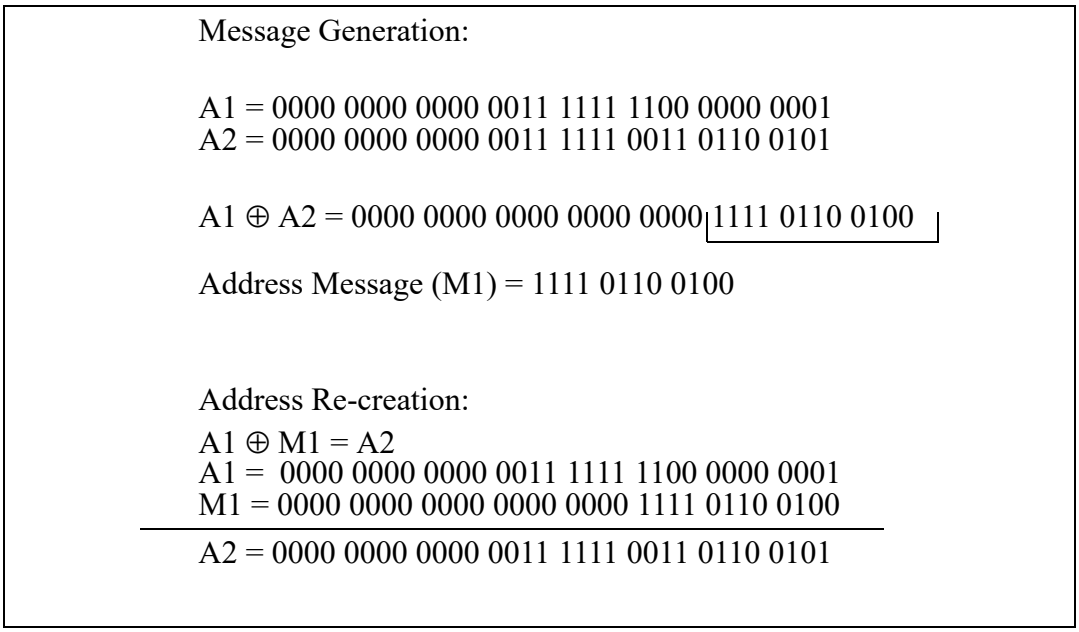
The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of ‘0’ indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of ‘1’ indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

Figure 1173. Relative address generation and recreation



66.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of ‘0’ indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of ‘1’ indicates that the preceding instruction count and history information should be interpreted in a VLE context.

66.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

66.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

66.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that 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 66.4.9: Nexus Overrun Control (OVCR) register*, for complete programming options.

66.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in *Section 66.8.1: Message Queue Overrun*. 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.

66.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstance:

- More than three messages are generated in the same cycle

Table 1114 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 that did not

successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in *Table 1114*.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (newer instruction) (WPM → DQM → OTM → BTM → DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 1114. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A(1)
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A(1)
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM — PID/NPIDR update (Program Correlation Message)	2	OTM Error Message
OT (Ownership)	OTM — PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message(2)
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM — Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

66.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

66.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

66.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

66.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

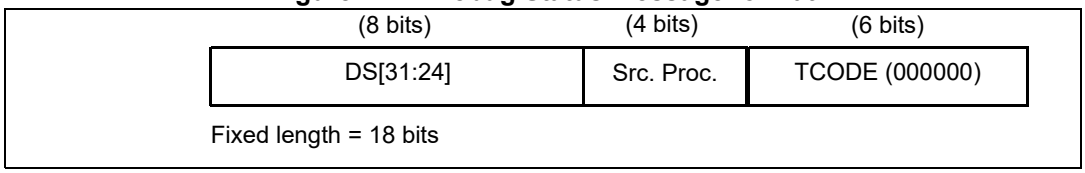
66.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

66.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

Figure 1174. Debug Status message format



66.10 Error messages

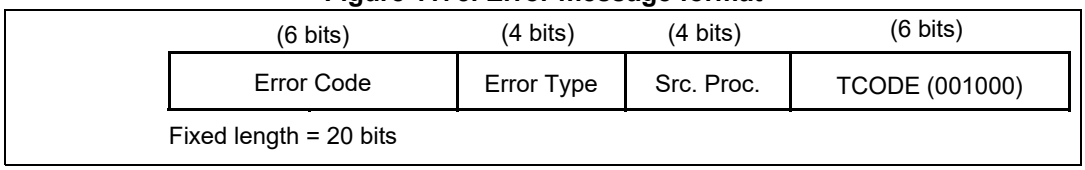
Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note: The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see *Table 1088* and *Table 1089*):

Figure 1175. Error message format



66.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

66.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

66.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z215An3) 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.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr**/**mtspr** instructions. This register is accessible in user or supervisor mode.

Nexus 3 Process ID

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

SPR — 517; Read/Write; Reset: 0

Figure 1176. Nexus 3 Process ID Register (NPIDR)

Note: OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.
- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value 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 1177. Ownership Trace Message format

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15–46 bits

66.12 Program Trace

This section details the program trace mechanism supported by Nexus3 for the e200z215An3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for Core (e200z215An3) processors is accomplished by snooping the internal address bus, attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

66.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

66.12.1.1 Indirect Branch Message instructions

Table 1115 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 1115. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr**, **bcctrl**, **bclr**, **bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	se_sc, tw
Return from interrupts / exceptions	se_rfi, se_rfci, se_rfdi
Exit from reset with Program Trace Enabled	Indirect branch with Sync, target address is initial instruction, count = 1

66.12.1.2 Direct Branch Message Instructions

Table 1116 shows the types of instructions that 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 1116. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b**, **ba**, **bl**, **bla**, **bc**, **bca**, **bcl**, **bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

66.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

66.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of “events” that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

66.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages — Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages — Indirect Branch History, and Indirect Branch History with Synchronization Messages.

66.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to ‘1’, indirect branch information is messaged out in the following format:

Figure 1178. Indirect branch message (history) format

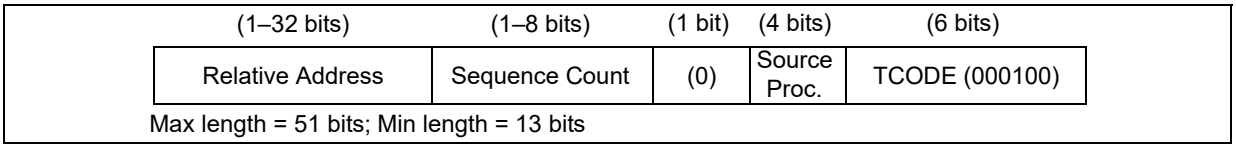
(1-32 bits)	(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Relative Address	Sequence Count	(0)	Source Proc.	TCODE (011100)

Max length = 83 bits; Min length = 14 bits

66.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to ‘0’, indirect branch information is messaged out in the following format:

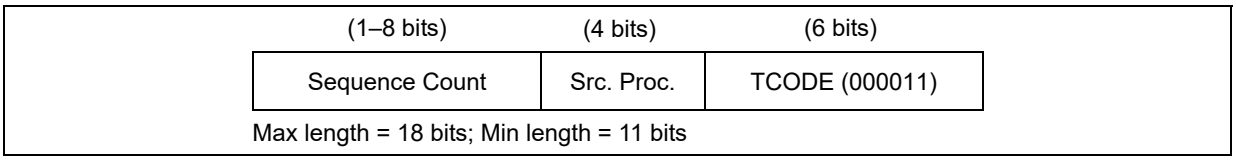
Figure 1179. Indirect branch message format



66.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

Figure 1180. 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.

66.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

66.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that 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 that trigger message generation are included in the ICNT. Instructions that 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.

66.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most

significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 1117. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**(1)

1. If the EVCODE for direct branch function calls is not masked in DC4, taken bl and bcl instructions will generate Program Correlation Messages and will not be logged in the history buffer.

66.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of ‘0’ indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of ‘1’ indicates that the preceding instruction count and history information should be interpreted in a VLE context.

66.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow; or, if program trace is in history mode, since the last instruction that recorded history information.

Figure 1181. Resource Full Message format

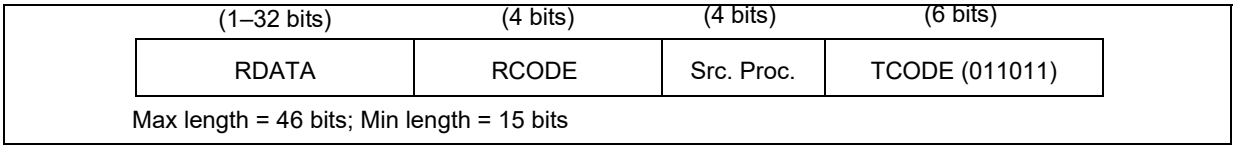


Table 1118 shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 1118. 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.

66.12.5 Program Correlation Messages

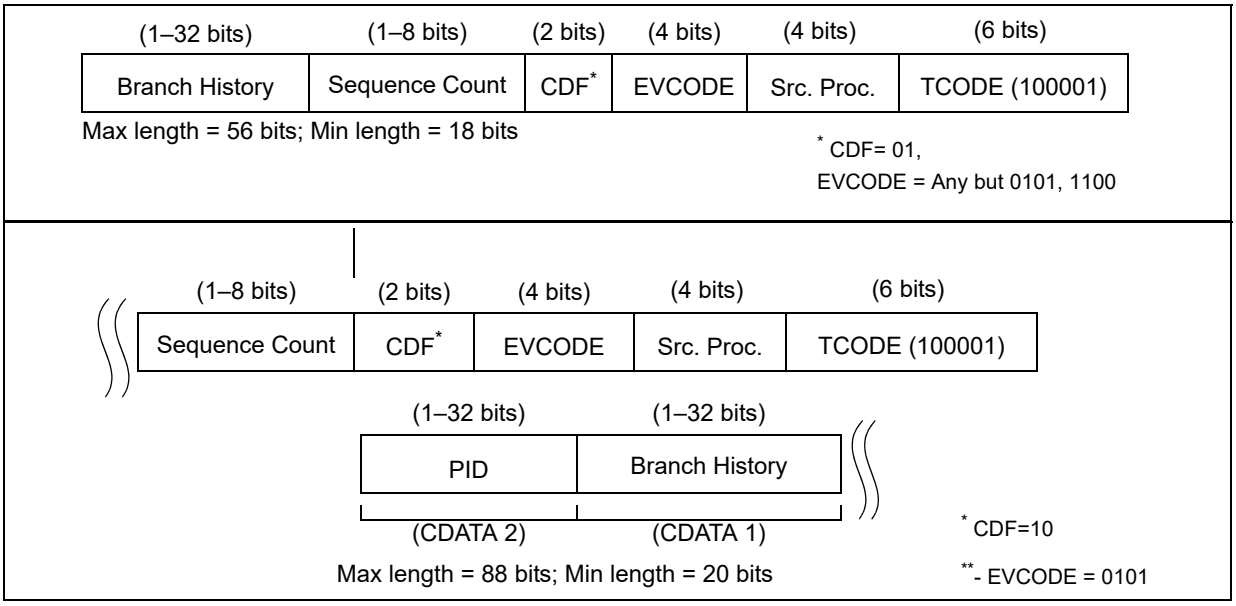
Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a “Branch and Link” instruction executes (direct branch function call — **bl**/**bcl**/**bla**/**bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}=‘0’ and $DC4_{PTMARK}$=‘1’.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to *Table 1091* 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 1182. Program Correlation message formats



66.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

66.12.6 Program Trace Overflow Error messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

66.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see

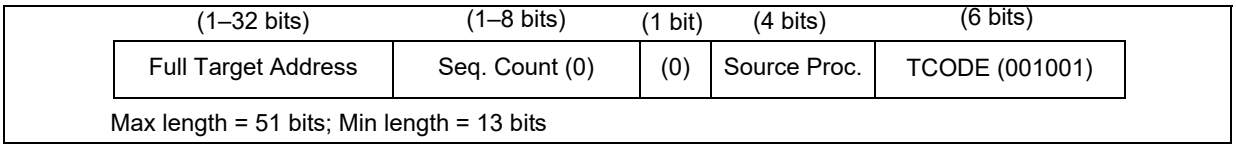
Table 1119):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow$ '1' with $DC4_{PTMARK}$='1'.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

Figure 1183. Program Trace Synchronization message format



Exception conditions that result in Program Trace Synchronization are summarized in *Table 1119*.

Table 1119. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first

instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

66.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see *Table 1119*):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync messages is as follows:

Figure 1184. Direct/Indirect Branch with Sync message format

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (001011 or 001100)

Max length = 51 bits; Min length = 13 bits

The format for Program Trace Indirect Branch History with Sync. Messages is as follows:

Figure 1185. Indirect Branch History w/ Sync message format

(1–32 bits)	(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (011101)

Max length = 83 bits; Min length = 14 bits

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in *Table 1120*.

Table 1120. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG}→OTM → BTM → DTM)→ Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

66.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Using the external hardware trace enable input signal (nex_ptm_starttr)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

66.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

Figure 1186. Program Trace — Indirect Branch message (Traditional)

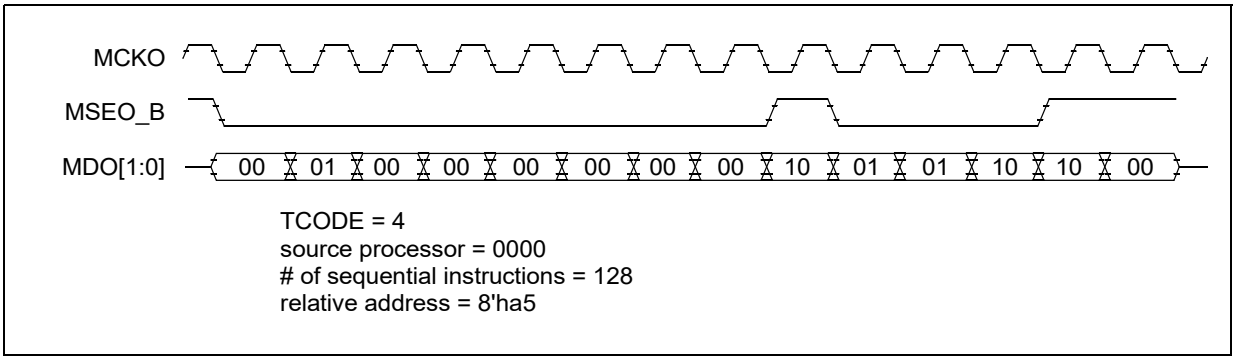


Figure 1187. Program Trace — Indirect Branch message (history)

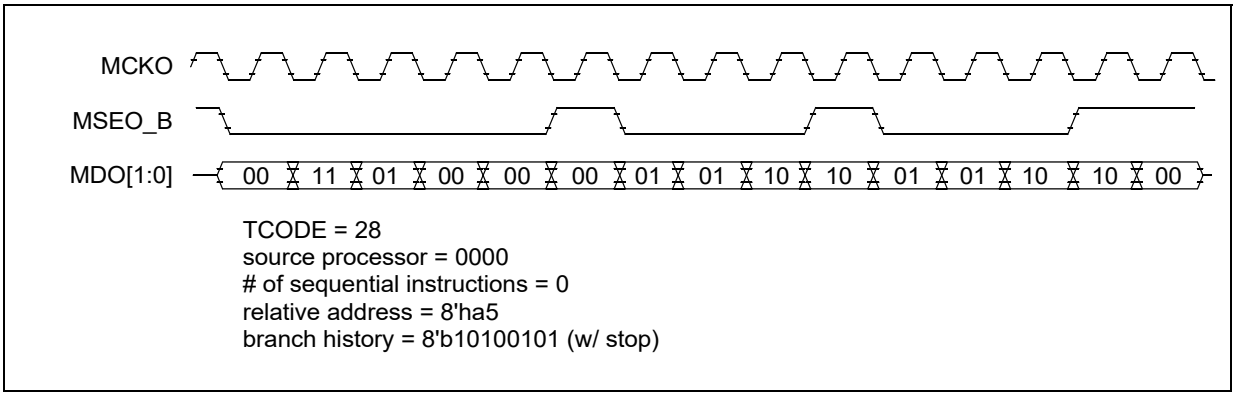


Figure 1188. Program Trace — Direct Branch (traditional) & Error messages

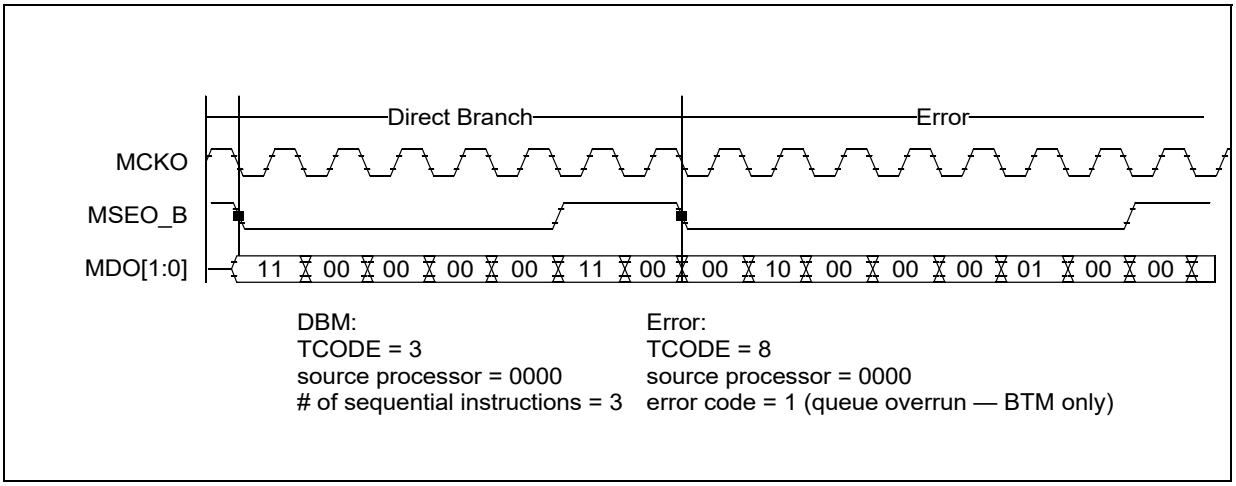
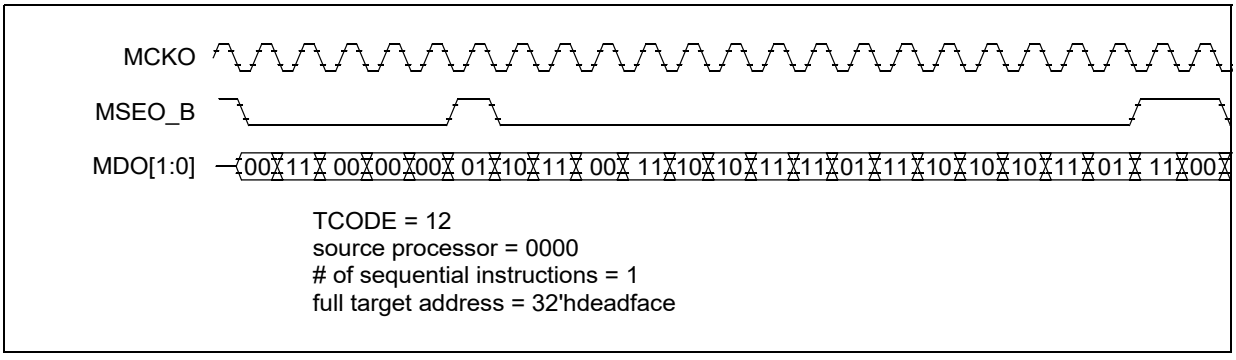


Figure 1189. Program Trace — Indirect Branch w/ Sync message



66.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

66.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note: Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z215An3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of three ways.

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).
- Using the external hardware trace enable signal (**nex_dtm_starttr**)

66.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization, and Error Messages.

66.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace message. Data Write message information is messaged out in the following format:

Figure 1190. Data Write message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

66.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

Figure 1191. Data Read message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

*Note: * Core (e200z215An3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.*

*Note: For Core (e200z215An3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.*

Note: The debug/development tool will need to distinguish the two cases based on the family of processor.

66.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see *Table 1121*):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

Figure 1192. Data Write/Read with Sync message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value	Full Address	Data Size	(0)	Source Proc.	TCODE (001101 or 001110)

Max length = 111 bits; Min length = 17 bits

Exception conditions that result in Data Trace Synchronization are summarized in *Table 1121*.

Table 1121. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.

Table 1121. Data Trace Exception summary(Continued)

Exception Condition	Exception Handling
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM)→ Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

66.13.3 DTM operation

66.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

66.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

66.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

66.13.3.4 Bus cycle special cases

Table 1122. Bus cycle special cases

Special case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$)(1)	Data Trace Message discarded

Table 1122. Bus cycle special cases(Continued)

Special case	Action
Bus cycle completed without error(1)	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) — both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note: For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (i.e. word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z215An3) Core Debug Support chapter.)

Note: A store to the store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

66.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

Figure 1193. Data Trace — Data Write message

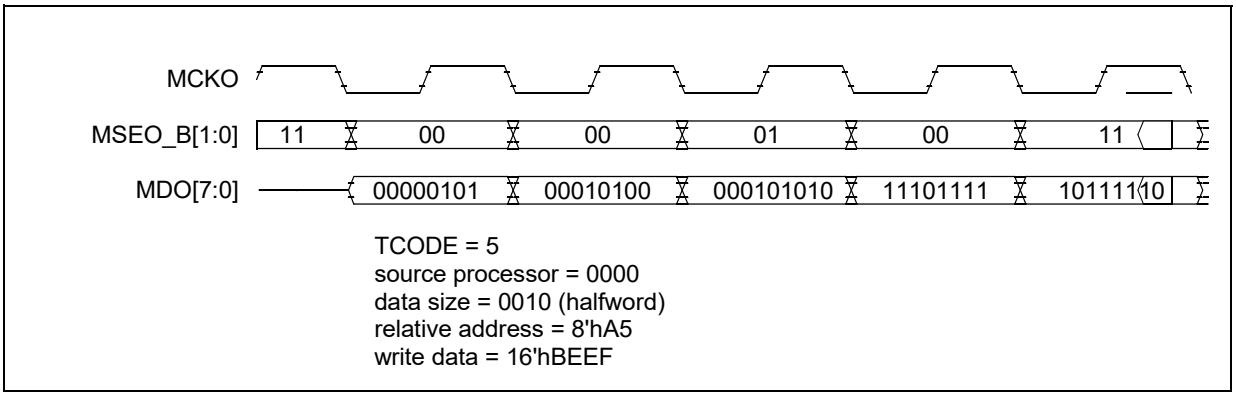
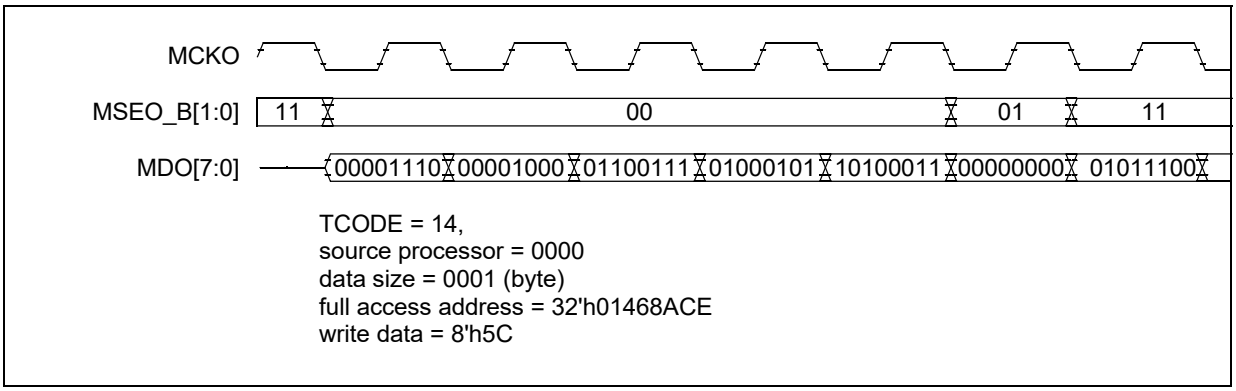


Figure 1194. Data Trace — Data Read w/ Sync message



66.14 Data Acquisition messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

66.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 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.

66.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

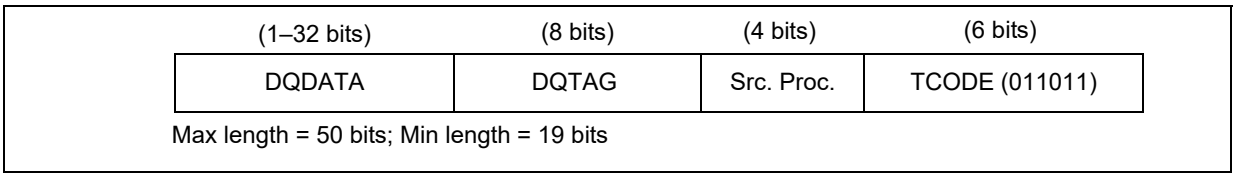
66.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the “External Debug Resource Allocation Control Register (EDBRAC0)” section in the Core (e200z215An3) 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 1195. Data Acquisition message format



66.15 Watchpoint Trace messaging

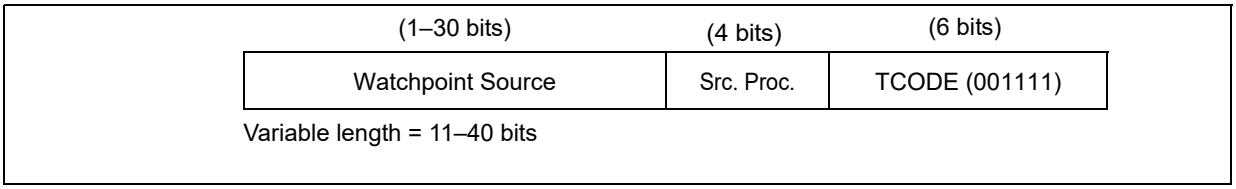
Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Refer to the Core (e200z215An3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**nex_evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See *Table 1126* for details on **nex_evto_b**.

Watchpoint information is messaged out in the following format:

Figure 1196. Watchpoint message format



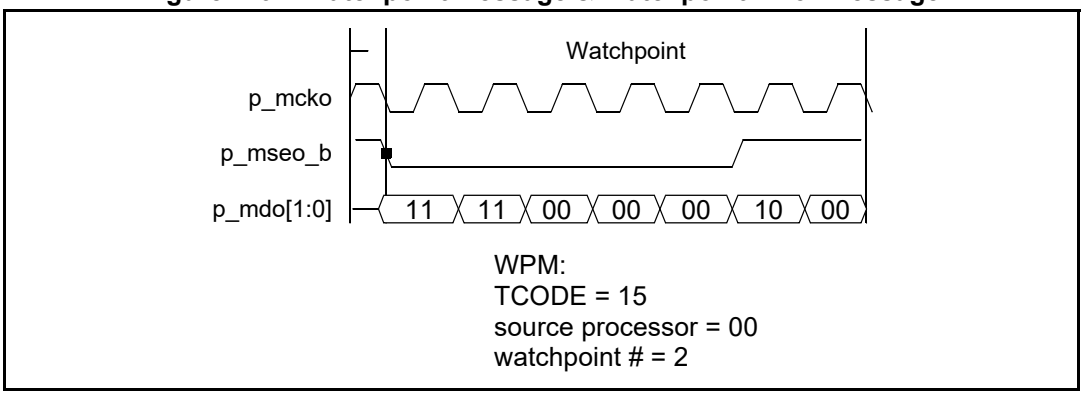
The Watchpoint Source message field will contain a ‘1’ for each asserted watchpoint. Leading zeros are truncated.

Table 1123. Watchpoint source encoding

Watchpoint source (1–30 bits)	Watchpoint description
00000000000000000000000000000000	No Watchpoints enabled for Watchpoint Trace Messaging
*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1	Watchpoint #0 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x*	Watchpoint #1 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx*	Watchpoint #2 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx*	Watchpoint #3 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx*	Watchpoint #4 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx*	Watchpoint #5 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx*	Watchpoint #6 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx*	Watchpoint #7 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx*	Watchpoint #8 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx*	Watchpoint #9 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx*	Watchpoint #10 enabled for WTM
*xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx*	Watchpoint #11 enabled for WTM
*xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx*	Watchpoint #12 enabled for WTM
*xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx*	Watchpoint #13 enabled for WTM
*xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx*	Watchpoint #14 enabled for WTM
*xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx*	Watchpoint #15 enabled for WTM
*xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #16 enabled for WTM
*xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #17 enabled for WTM
*xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #18 enabled for WTM
*xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #19 enabled for WTM
*xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #20 enabled for WTM
*xxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #21 enabled for WTM
*xxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #22 enabled for WTM
*xxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #23 enabled for WTM
*xxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #24 enabled for WTM
*xxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #25 enabled for WTM
*xxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #26 enabled for WTM
*xxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #27 enabled for WTM
*xxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #28 enabled for WTM
*xx*1*xxxxxxxxxxxxxxxx*	Watchpoint #29 enabled for WTM
*x*1*xxxxxxxxxxxxxxxx*	
1*xxxxxxxxxxxxxxxx*	

66.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

Figure 1197. Watchpoint Message & Watchpoint Error message



66.16 External Hardware Trigger Controls

The Nexus 3 module provides a set of hardware trigger inputs to allow for external hardware to also independently control enabling/disabling of Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace messaging. These signals can be used to produce a “window” for trace messaging for each type.

When a start trigger condition is detected for one of the trace types, and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace features become enabled, and the corresponding enable bit(s) of the DC1 register is set. When a stop trigger condition is detected and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace feature becomes disabled, and the corresponding enable bit of the DC1 register is cleared. If a simultaneous trigger condition is received for both start and stop triggering, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger condition occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature control action that is derived from external hardware triggering, and if the corresponding trace feature has been enabled by a direct write to DC1, then the corresponding hardware trigger signals will have no effect. *Table 1124* details the external hardware trigger signals.

Table 1124. External Hardware Trigger Signals

Name	Description
nex_dtm_starttr	Data Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_dtm_stoptr	Data Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_otm_starttr	Ownership Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_otm_stoptr	Ownership Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_ptm_starttr	Program Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_ptm_stoptr	Program Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_wtm_starttr	Watchpoint Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_wtm_stoptr	Watchpoint Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.

These signals are active-high inputs and are transition sensitive. A 0→1 transition on the signal indicates the corresponding condition is requested. These signals are not synchronized, and must meet setup and hold requirements relative to **nex_clk**.

66.17 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6‘b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note: Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

66.17.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 66.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) Write Address → 32h’xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in *Section 66.5: Nexus 3 Register Access via JTAG/OnCE*. Configure the bits as follows:
   a) Access Control (AC) → 1b’1 (to indicate start access)
   b) Map Select (MAP) → 3b’000 (primary memory map)
   c) Access Priority (PR) → 2b’11 (highest priority)
   d) Read/Write (RW) → 1b’1 (write access)
   e) Word Size (SZ) → 3b’0xx (32-bit, 16-bit, 8-bit)
   f) 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 66.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) 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). The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. 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 1126* for details) and clears the ERR bit in the RWCS Register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once DV has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

*Note: Only the **nex_ahb_start**, **nex_rdy_b**, and **nex_err_b** pins as well as the AC, DV, and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

66.17.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in *Section 66.17.1: Single write access* 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). The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and the nex_rdy_b pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS Register, otherwise the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Once DV has been cleared, this indicates that the device is ready for the next access in the block transfer, or an error has occurred.
3. If AC has not been cleared due to an error, repeat Step 3 in *Section 66.17.1: Single write access* 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.

66.17.3 Single read access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 66.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in *Section 66.5: Nexus 3 Register Access via JTAG/OnCE*. Configure the bits as follows:
   a) Access Control (AC) → 1b'1 (to indicate start access)
   b) Map Select (MAP) → 3b'000 (primary memory map)
   c) Access Priority (PR) → 2b'11 (highest priority)
   d) Read/Write (RW) → 1b'0 (read access)
   e) Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   f) 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. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. 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 1126* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in *Section 66.5: Nexus 3 Register Access via JTAG/OnCE*.

*Note: Only the **nex_ahb_start**, **nex_rdy_b**, and **nex_err_b** pins as well as the AC, DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

66.17.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in *Section 66.17.3: Single read access* 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 1126* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS Register, and asserts the **nex_rdy_b** pin (see *Table 1126* for detail on **nex_rdy_b**).

Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.

4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in *Section 66.5: Nexus 3 Register Access via JTAG/OnCE*.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in *Section 66.17.3: Single read access* 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).

Note: 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.

66.17.5 Error handling

The Nexus 3 module handles various error conditions as follows:

66.17.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (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.

66.17.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

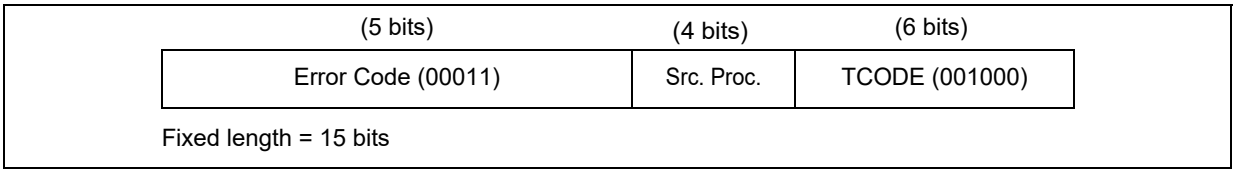
1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
   a) If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
   b) If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

66.17.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

Figure 1198. Error message format



66.18 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

66.18.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **nex_evto_b** pin, (4) **nex_wevto[3:0]** pins, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in *Table 1126*. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see *Section 66.5: Nexus 3 Register Access via JTAG/OnCE* 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 (e200z215An3) Core Debug Support chapter.

Table 1125. JTAG pins for Nexus 3

JTAG pins	Input/ Output	Description of JTAG pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the “Shift-IR” and “Shift-DR” controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in *Table 1126*.

Table 1126. Nexus 3 auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0])** are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains deasserted
nex_err_b	O	Error (**nex_err_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, **nex_err_b** remains deasserted
nex_evto_b	O	Event Out (**nex_evto_b**) is an output that, 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 that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1) Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2’b00). 2) Debug request to Nexus1 module (provided EIC = 2’b01 and this feature is implemented).
nex_wevto[3:0]	O	Watchpoint Event Out 3:0 (**nex_wevto[3:0]**) are outputs that, when asserted, indicates one or more watchpoint events has occurred based on the settings in the DC2 and DC3 registers. **nex_wevto[3:0]** is held asserted for one (1) cycle of **nex_mcko**.
nex_ext_src_id[0:3]	I	nex_ext_src_id[0:3] is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time
nex_sfwcntl_en]	I	nex_sfwcntl_en is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to *Section 66.20: Auxiliary port arbitration*, for detail on Nexus port arbitration.

Table 1127. 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 1129*.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant (**nar_aux_grant**) is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

66.18.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in *Table 1128*. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

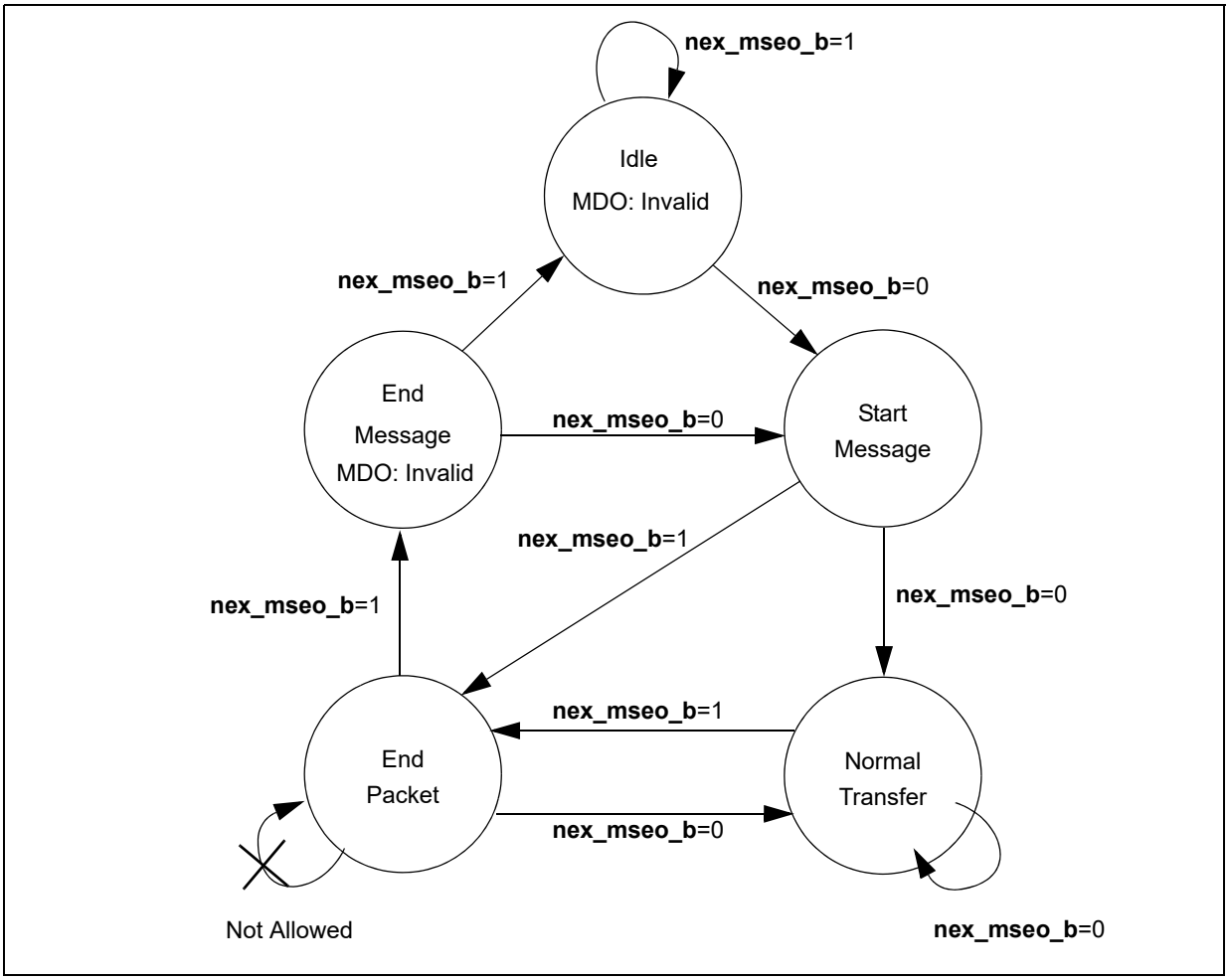
Single pin MSEO is not supported on the SPC572Lx.

Table 1128. 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 1199 illustrates the state diagram for single pin MSEO transfers.

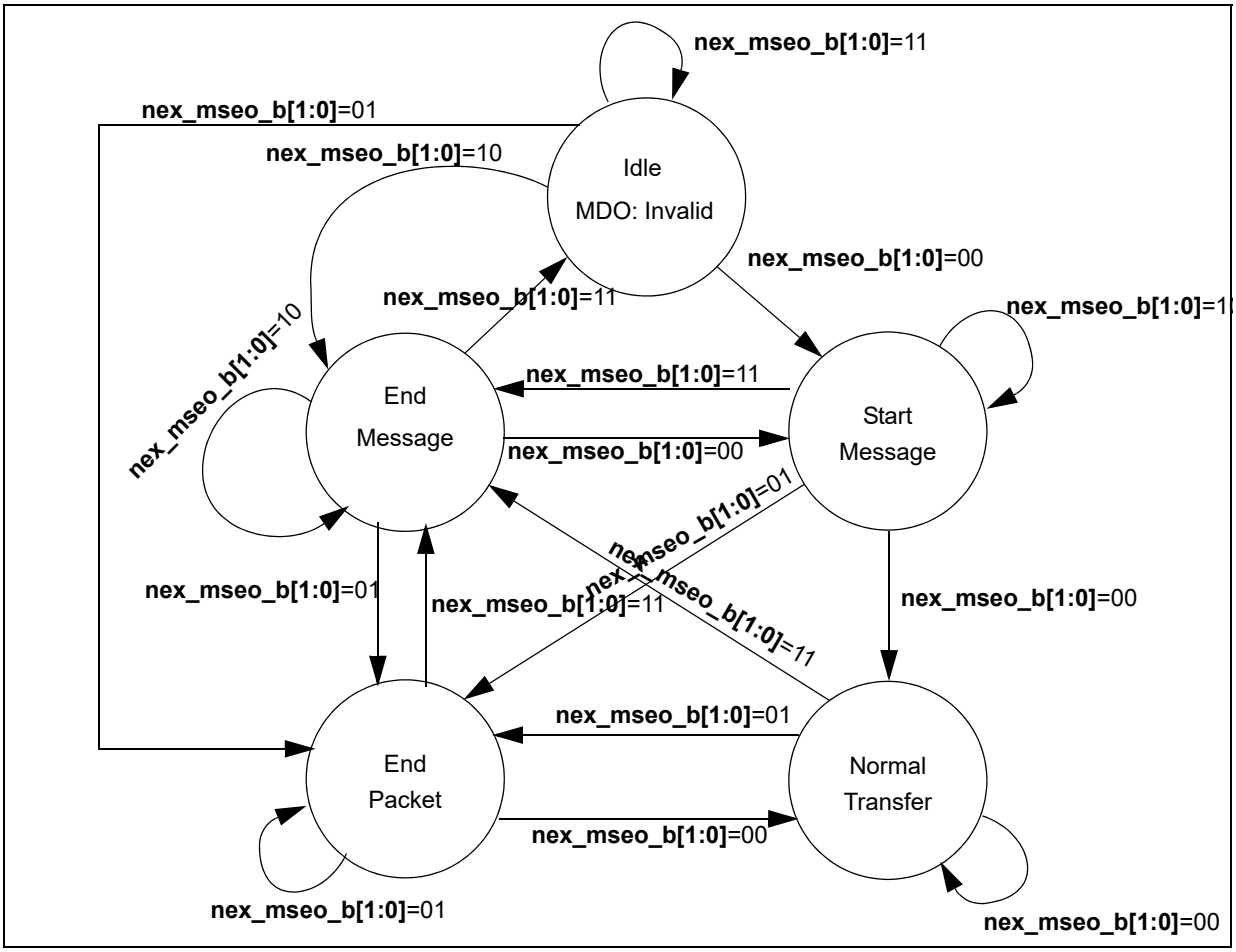
Figure 1199. Single pin MSEO transfers



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 1200 illustrates the state diagram for dual pin MSEO transfers.

Figure 1200. Dual pin MSEO transfers



The dual pin MSEO option is more robust than 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.

66.19 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bit is from the last packet and which bit is from the next packet.)
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-

order bit so that it can end on a port boundary.

For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

66.20 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in *Table 1129* below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in *Section 66.18.2: Pin protocol*. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the MSEO state machine reaches the “End Message” state.

Table 1129. 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

66.21 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 1130 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. *Table 1131* illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 1130. 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 1131. 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 1132 and *Table 1133* 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 1132. 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 1133. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 1134 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and *Table 1135* illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 1134. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 1135. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

66.22 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, refer to the SPC572Lx Data Sheet.

66.23 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

66.23.1 JTAG sequence for accessing internal Nexus registers

Table 1136. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE → SELECT-DR_SCAN
2	0	SELECT-DR_SCAN → CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR → SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR → EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR → UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR → SELECT-DR_SCAN
8	0	SELECT-DR_SCAN → CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR → SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR → EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR → UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR → RUN-TEST/IDLE (transfer complete — Nexus controller to Reg. Select state)

66.23.2 JTAG sequence for read access of memory-mapped resources

Table 1137. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

66.23.3 JTAG sequence for write access of memory-mapped resources

Table 1138. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value — data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address — data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
8	—	If CNT > 0, go back to Step #5

67 Register Protection (REG_PROT)

67.1 Introduction

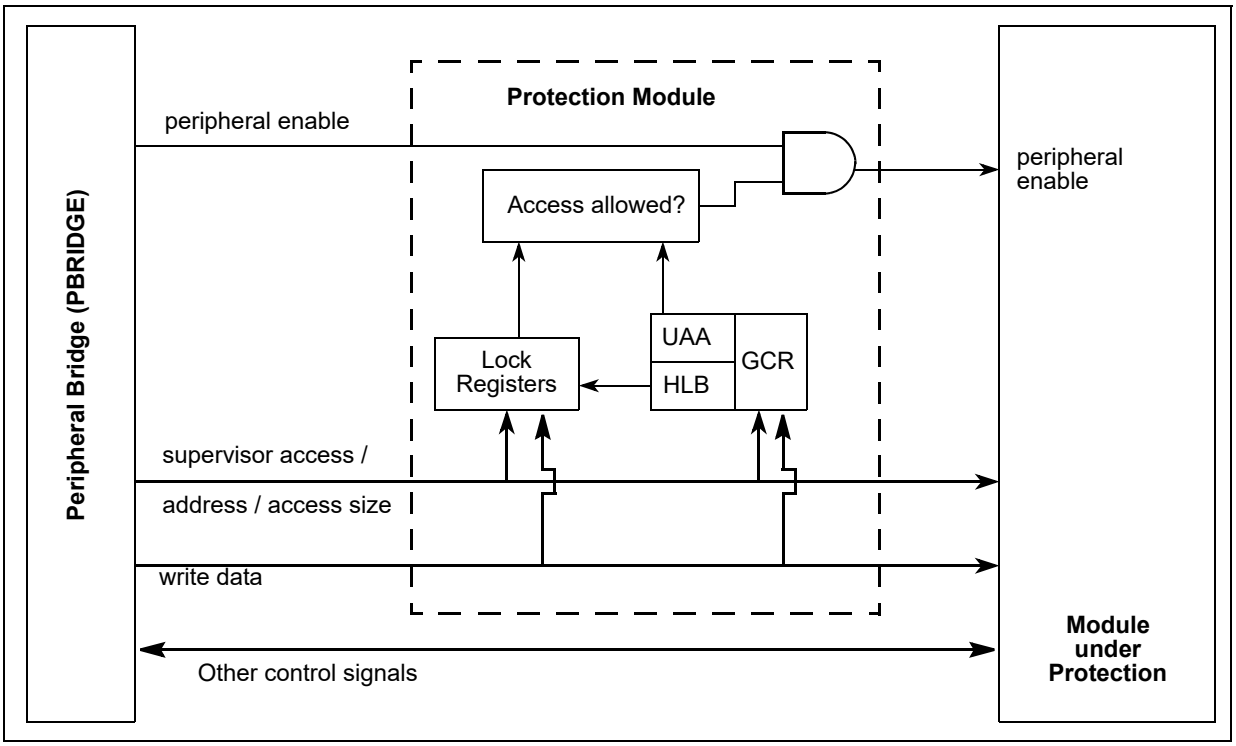
67.1.1 Overview

The Register Protection (REG_PROT) module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

Note: See the device reference manual 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 1201*.

Figure 1201. REG_PROT Block Diagram



67.1.2 Features

Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Write to address mirror automatically sets corresponding lock bit
- Once configured lock bits can be protected from changes

67.1.3 Modes of operation

The Register Protection module is operable when the module under protection is operable.

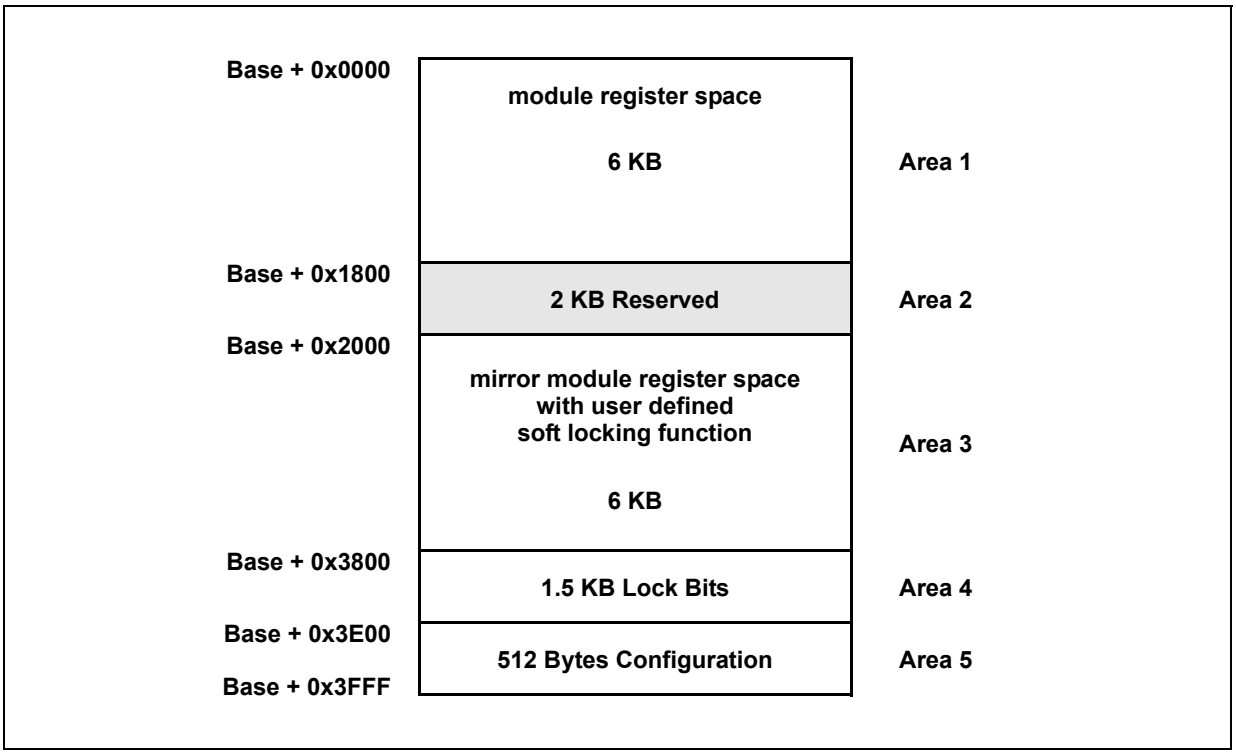
67.2 External signal description

There are no external signals.

67.3 Memory map and register definition

This section provides a detailed description of the memory map of a module with register protection. The original 16 KB module memory space is divided into five areas as shown in *Figure 1202*.

Figure 1202. REG_PROT Memory Diagram



- Area 1 is 6 KB and holds the normal functional module registers and is transparent for all read/write operations.
- Area 2 is 2 KB starting at offset 0x1800 is a reserved area, which shall not be accessed.
- Area 3 is 6 KB, starting at offset 0x2000 and is a mirror of area 1. A read/write access to an offset 0x2000+X will read/write the register at offset X. However a write access to offset 0x2000+X will additionally set the optional Soft Lock Bits for this offset X in the same cycle as the register at offset X is written. This provides for an atomic write and lock operation. Not all registers in area 1 need to have protection defined by associated

Soft Lock Bits. For unprotected registers at offset Y, accesses to offset 0x2000+Y will be identical to accesses at offset Y.

- Area 4 is 1.5 KB and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.
- Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-bit registers in Areas 4 and 5 will result in a transfer error.

67.3.1 Memory map

Figure 1139 shows the memory map for a module which is protected via a REG_PROT module.

Table 1139. REG_PROT Memory Map

Offset	Use	Location
0x0000	Module Register 0 (MR0)	*on page 1948*
0x0001	Module Register 1 (MR1)	*on page 1948*
0x0002	Module Register 2 (MR2)	*on page 1948*
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)	*on page 1948*
0x1800 - 0x1FFF	Reserved	
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)	*on page 1948*
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)	*on page 1948*
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)	*on page 1948*
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3	*on page 1948*
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7	*on page 1948*
0x3802 - 0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 - Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143	*on page 1948*
0x3E00 - 0x3FFB	Reserved	
0x3FFC	Global Configuration Register (GCR)	*on page 1950*

Note: Reserved registers in area #1 will be handled as specified in the protected module's documentation.

67.3.2 Register descriptions

This section describes in address order all the REG_PROT registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

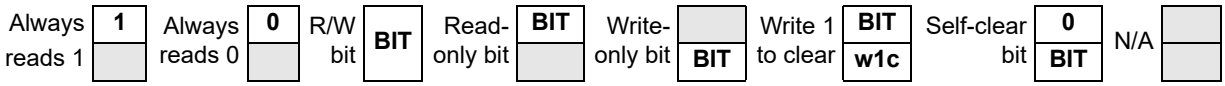


Figure 1203. Key to Register Fields

67.3.2.1 Module Registers (MR0-6143)

This is the lower 6 KB module memory space which holds all the functional registers of the module that is protected by the REG_PROT module.

67.3.2.2 Module Register and Set Soft Lock Bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBR*n*.SLB*m*, according to the mapping described in *Table 1141*.

67.3.2.3 Soft Lock Bit Register (SLBR0-1535)

The Soft Lock Bit Registers hold the Soft Lock Bits for the protected registers in memory area #1, which is the normal register address space of the protected module. Each SLB register has a four Soft Lock Bits (SLB)-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 1140. 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
...	...

- 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.

Address 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 1204. Soft Lock Bit Register (SLBR*n*)

Table 1141. 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

Table 1141. SLBR*n* field descriptions(Continued)

Field	Description
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

67.3.2.4 Global Configuration Register (GCR)

This register controls the configuration of the locking of the SLB's and the level of user access allowed to the protected registers.

Note: Access in User Mode is read-only; in Supervisor Mode access is read/write.

Offset 0x3FFC

Access: Read Always
Supervisor write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HLB	0	0	0	0	0	0	0	UAA	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1205. Global Configuration Register (GCR)

Table 1142. GCR field descriptions

Field	Description
0 HLB	Hard Lock Bit. This register cannot 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 cannot be modified 0 All SLB bits are accessible and can be modified.
1 - 7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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.
9 - 31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Note: The GCR.UAA bit has no effect on the allowed access modes for the registers in the REG_PROT module.

67.4 Functional description

The following sections describe the functional characteristics of the Register Protection module.

67.4.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)

Which addresses are protected and the protection size depends on the SoC and/or module.

For all addresses that are protected there are SLBR*n*.SLB*m* bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding SLBR*n*.SLB*m* bit is always 0b0 no matter what software is writing to.

67.4.2 Change lock settings

To change the setting whether an address is locked or unlocked the corresponding SLBR*n*.SLB*m* bit needs to be changed. This can be done using the following methods:

- Modify the SLBR*n*.SLB*m* directly by writing to area #4
- Set the SLBR*n*.SLB*m* bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

67.4.2.1 Change lock settings directly via Area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each SLBR*n*.SLB*m* bit has a mask bit SLBR*n*.WE*m* which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

Figure 1206 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 1206. Change lock settings directly via Area #4

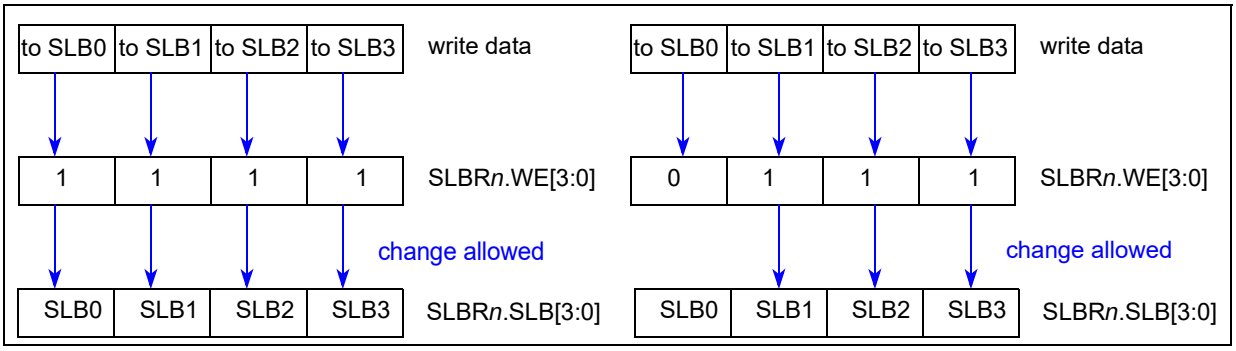
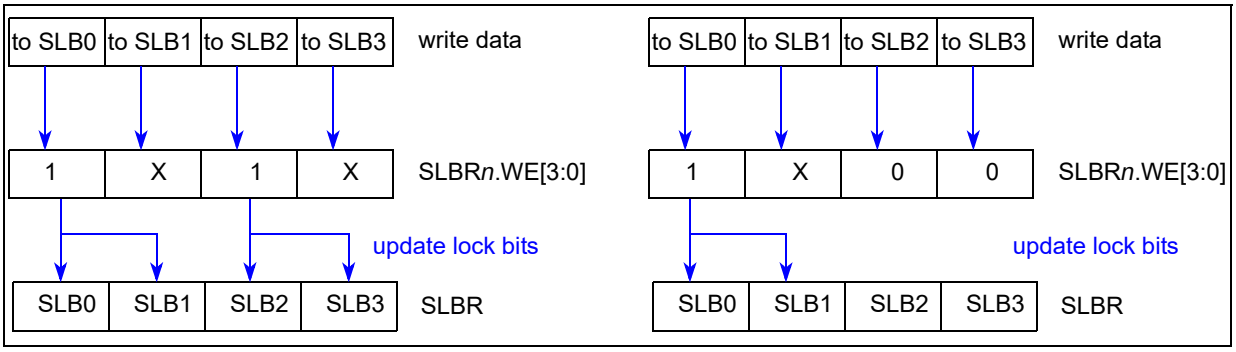


Figure 1206 shows four registers that can be protected 8-bit wise. In *Figure 1207* registers with 16-bit protection and in *Figure 1208* registers with 32-bit protection are shown:

Figure 1207. Change lock settings for 16-bit protected addresses

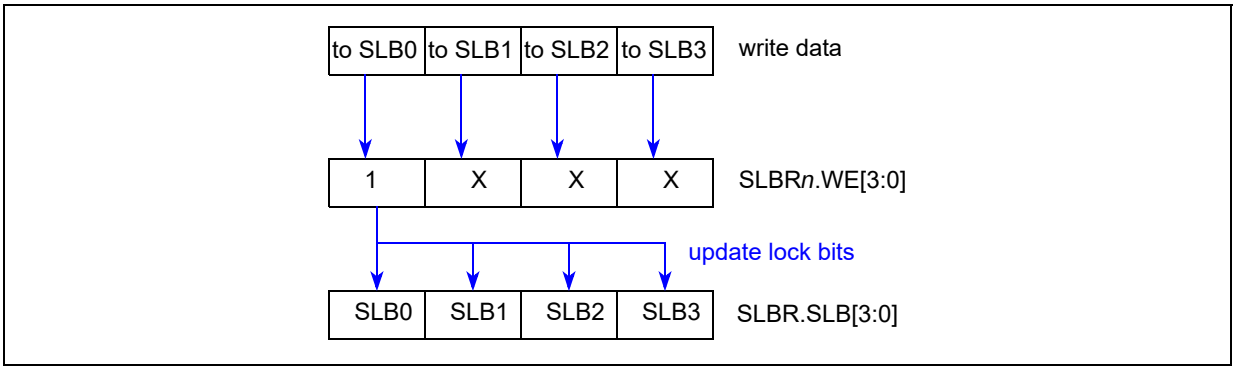


On the right side of *Figure 1207* it is shown that the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[1] also. This is done as the address reflected by SLBR*n*.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBR*n*.WE[0] must be set while SLBR*n*.WE[1] 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 1207* 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.

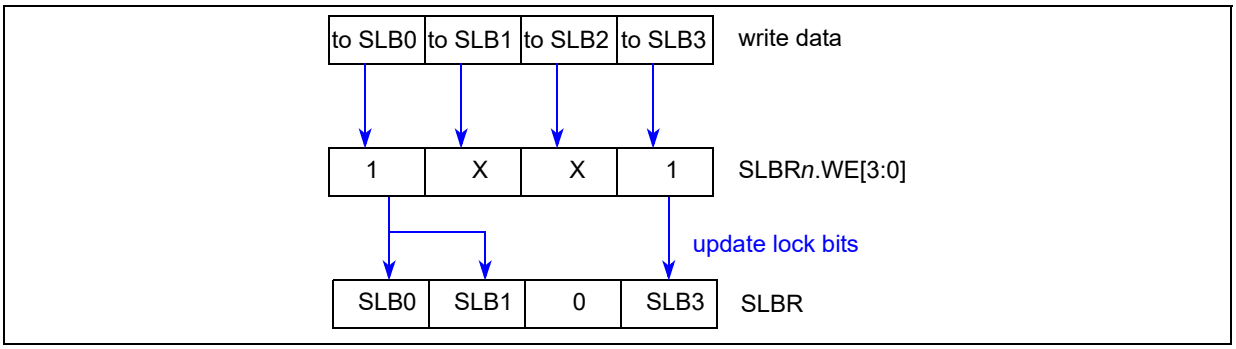
In *Figure 1208* a 32-bit wise protected register is shown. When SLBR*n*.WE[0] is set the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[3:1] also. Otherwise SLBR*n*.SLB[3:0] remains unchanged.

Figure 1208. Change lock settings for 32-bit protected addresses



In *Figure 1209* an example is shown which has a mixed protection size configuration:

Figure 1209. Change lock settings for mixed protection

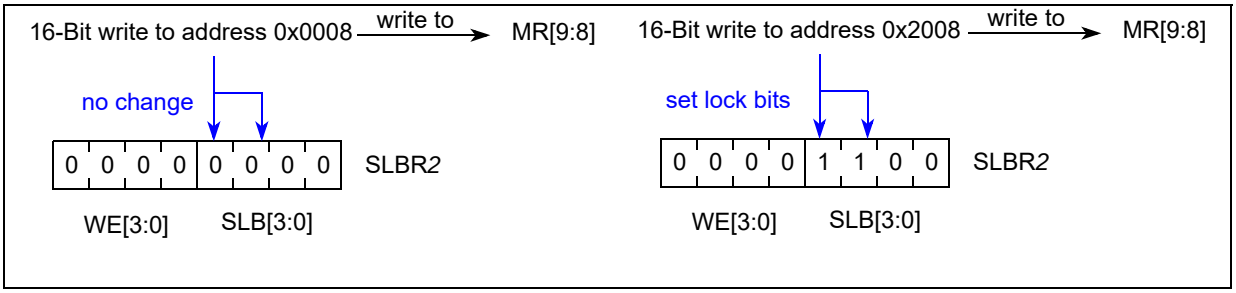


The data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBR*n*.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBR*n*.SLB[3] is written to SLBR*n*.SLB[3].

67.4.2.2 Enable locking via mirror module space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. *Figure 1210* shows one example:

Figure 1210. Enable Locking Via Mirror Module Space (Area #3)

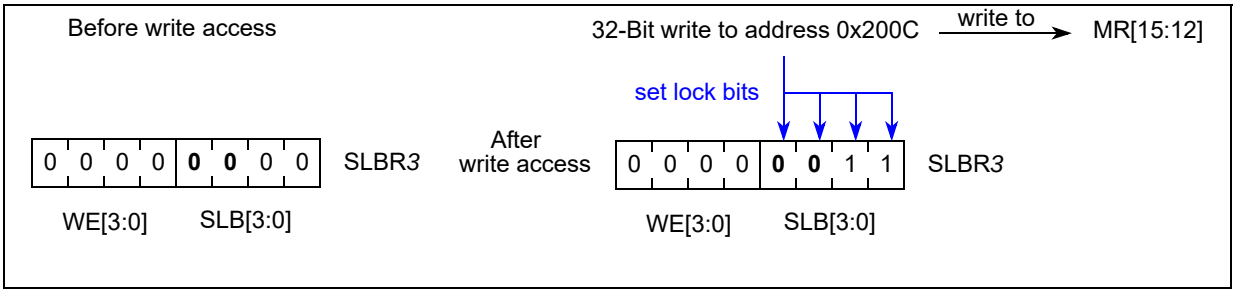


When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (left part of *Figure 1207*).

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 1207*).

Figure 1211 shows an example where some addresses are protected and some are not:

Figure 1211. Enable locking for protected and unprotected addresses



In the example in *Figure 1211* addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

Note: Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

67.4.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in *Section 67.4.2.1, Change lock settings directly via Area #4* and *Section 67.4.2.2, 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.

67.4.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to *Figure 1202*

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.
3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.
6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set

67.5 Initialization/application information

67.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (See *Section 67.3.2, Register descriptions*). In summary, after reset, locking for all MR*n* registers is disabled. The registers can be accessed in Supervisor Mode only.

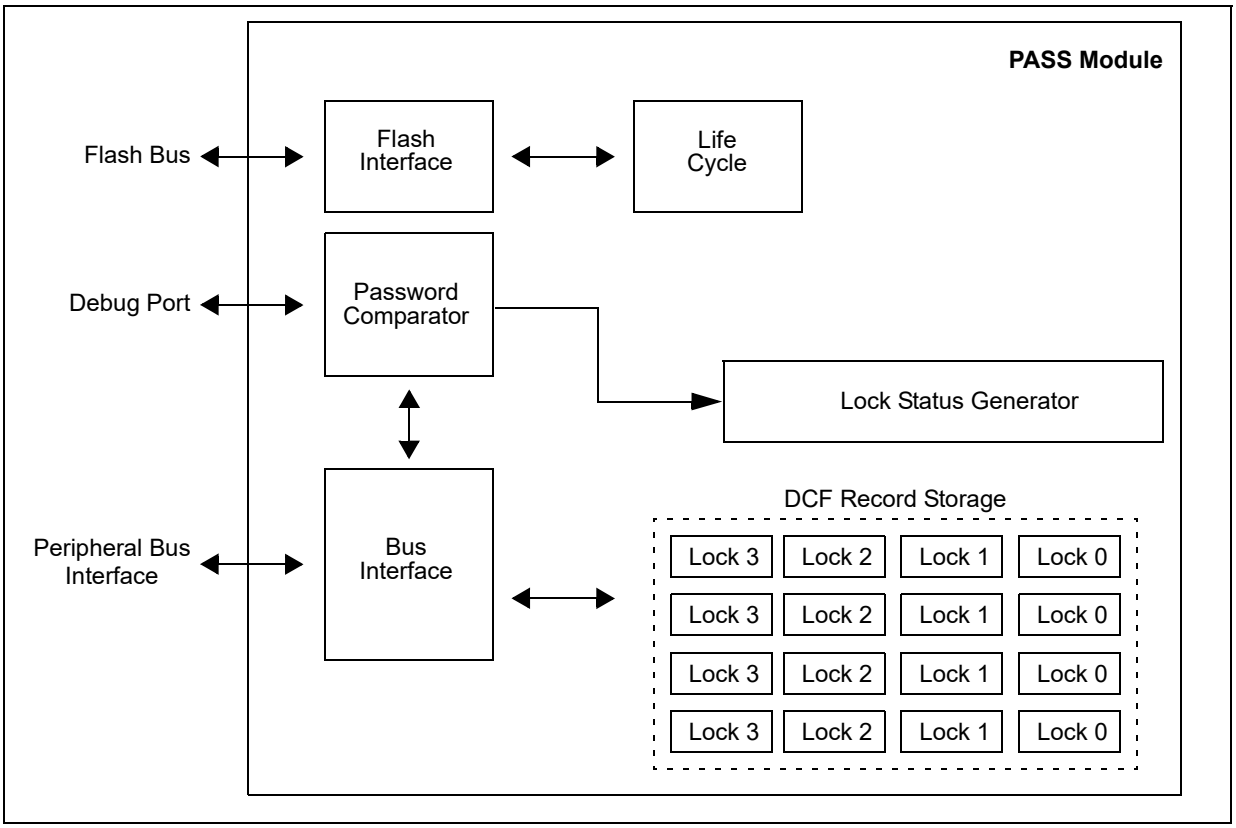
68 Password and Device Security Module (PASS)

68.1 Introduction

68.1.1 Overview

The Password and device security (PASS) module shown in *Figure 1212*, receives password challenges and determines their validity. It also maintains the device security and access states.

Figure 1212. PASS module block diagram



68.1.2 Features

The PASS includes these distinctive features:

- Life cycle status
- Password comparison
- JTAG password comparison

68.1.3 Modes of operation

The PASS operates identically in all system modes.

68.2 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the PASS. (DCF clients are described in *Section 68.3, DCF clients*.)

Table 1143 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 of the Reference Manual .

Do not write to address 0x00D0 with bit value 0.

Table 1143. PASS memory map

Offset	Register	Access	Reset Value	DCF Client Address (ADDR[14:0])	Section/Page
0x0000	Life Cycle Status Register (LCSTAT)	R	0x0000_00xx(1)	None	*68.2.1.1/1958*
0x0008	Challenge Selector Register (CHSEL)	R/W	0x0000_0000	None	*68.2.1.2/1959*
0x0010	Challenge Status Register (CSTAT)	R	0x0000_0000	None	*68.2.1.3/1960*
0x0020	Challenge Input Register 0 (CIN0)	W	0x0000_0000	None	*68.2.1.4/1960*
0x0024	Challenge Input Register 1 (CIN1)	W	0x0000_0000	None	*68.2.1.4/1960*
0x0028	Challenge Input Register 2 (CIN2)	W	0x0000_0000	None	*68.2.1.4/1960*
0x002C	Challenge Input Register 3 (CIN3)	W	0x0000_0000	None	*68.2.1.4/1960*
0x0030	Challenge Input Register 4 (CIN4)	W	0x0000_0000	None	*68.2.1.4/1960*
0x0034	Challenge Input Register 5 (CIN5)	W	0x0000_0000	None	*68.2.1.4/1960*
0x0038	Challenge Input Register 6 (CIN6)	W	0x0000_0000	None	*68.2.1.4/1960*
0x003C	Challenge Input Register 7 (CIN7)	W	0x0000_0000	None	*68.2.1.4/1960*
Password Group 0					
0x0100	Lock 0 Status Register (LOCK0_PG0)	R/W	0xxxxx_xxxx(2)	0x0100	*68.2.1.5/1961*
0x0104	Lock 1 Status Register (LOCK1_PG0)	R/W	0xx000_xxxx	0x0104	*68.2.1.6/1963*
0x0108	Lock 2 Status Register (LOCK2_PG0)	R/W	0xxxxx_xxxx(2)	0x0108	*68.2.1.7/1964*
0x010C	Lock 3 Status Register (LOCK3_PG0)	R/W	0xxxxx_xxxx(2)	0x010C	*68.2.1.8/1965*
Password Group 1					
0x0110	Lock 0 Status Register (LOCK0_PG1)	R/W	0xxxxx_xxxx(2)	0x0110	*68.2.1.5/1961*
0x0114	Lock 1 Status Register (LOCK1_PG1)	R/W	0xx000_xxxx	0x0114	*68.2.1.6/1963*
0x0118	Lock 2 Status Register (LOCK2_PG1)	R/W	0xxxxx_xxxx(2)	0x0118	*68.2.1.7/1964*
0x011C	Lock 3 Status Register (LOCK3_PG1)	R/W	0xx0xx_xxxx(2)	0x011C	*68.2.1.8/1965*
Password Group 2					
0x0120	Lock 0 Status Register (LOCK0_PG2)	R/W	0xxxxx_xxxx(2)	0x0120	*68.2.1.5/1961*

Table 1143. PASS memory map(Continued)

Offset	Register	Access	Reset Value	DCF Client Address (ADDR[14:0])	Section/Page
0x0124	Lock 1 Status Register (LOCK1_PG2)	R/W	0xx000_xxxx	0x0124	*68.2.1.6/1963*
0x0128	Lock 2 Status Register (LOCK2_PG2)	R/W	0xxxxx_xxxx(2)	0x0128	*68.2.1.7/1964*
0x012C	Lock 3 Status Register (LOCK3_PG2)	R/W	0xx0xx_xxxx(2)	0x012C	*68.2.1.8/1965*
Password Group 3					
0x0130	Lock 0 Status Register (LOCK0_PG3)	R/W	0xxxxx_xxxx(2)	0x0130	*68.2.1.5/1961*
0x0134	Lock 1 Status Register (LOCK1_PG3)	R/W	0xx000_xxxx	0x0134	*68.2.1.6/1963*
0x0138	Lock 2 Status Register (LOCK2_PG3)	R/W	0xxxxx_xxxx(2)	0x0138	*68.2.1.7/1964*
0x013C	Lock 3 Status Register (LOCK3_PG3)	R/W	0xx0xx_xxxx(2)	0x013C	*68.2.1.8/1965*

1. Reset value depends on life cycle status
2. Reset value depends on password group lock state

Note: Non-implemented registers generate bus aborts if enabled by the SOC.

68.2.1 Register descriptions

The following registers are available in the PASS. The shaded bits are reserved for future use. For future compatibility, only write ‘0’ to these bits; they always read ‘0’.

68.2.1.1 Life Cycle Status register (LCSTAT)

The Life Cycle Status Register reflects the production status and censorship status of the device. It is possible to mature the device (move the device life cycle forward), but never to revert the life cycle to an earlier state.

The reset value of the LCSTAT register LIFE field derives from an external life cycle management module (like SSCM). It is a 3-bit life cycle. This register simply displays the life cycle 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	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	x(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	LIFE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 1213. Life Cycle Status register (LCSTAT)

1. Reset value depends on the Censorship DCF client

PASS_LCSTAT register functions are as shown in *Table 1144*.

Table 1144. 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 *Section 68.5.4, Censoring and uncensoring the device*. 1 Device is censored 0 Device is uncensored
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.
2	Reserved. This read-only bit is reserved and always has the value of 0.
3–28	Reserved. This read-only bit is reserved and always has the value of 0.
29-–31 LIFE	The life cycle setting is manipulated externally (SSCM). Pass displays a 3-bit input life cycle. 011 Customer Delivery 010 OEM Production 110 ST Production 111 In Field 001 reserved 000 Failure Analysis mode

68.2.1.2 Challenge Selector register (CHSEL)

This register determines which password group is challenged via the Challenge Input 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 1214. CHSEL register

CHSEL register functions are as shown in *Table 1145*.

Table 1145. CHSEL register field descriptions

Field	Description
0–29	This read-only bit is reserved and always has the value of 0.
30–31 GRP	Password Group to unlock 00Password Group 0 01 Password Group 1 10 Password Group 2 11 Password Group 3

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.

68.2.1.3 Challenge Status register (CSTAT)

This register provides status information for the password challenge.

Offset 0x0010 Access: Software Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	CMST			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1215. CSTAT register

CSTAT register functions are as shown in *Table 1146*.

Table 1146. CSTAT register field descriptions

Field	Description
0–27	This read-only bit is reserved and always has the value of 0.
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 results in a comparison fail regardless of the values written.

68.2.1.4 Challenge Input register (CIN*n*)

This register is used to provide the Challenge word *n* for a password group as selected by the Challenge Selector Register.

Offset 0x0020 CIN0
0x0024 CIN1
0x0028 CIN2
0x002C CIN3
0x0030 CIN4
0x0034 CIN5
0x0038 CIN6
0x003C CIN7

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	PW32															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	PW32															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1216. CIN*n* register

CIN register functions are as shown in *Table 1147*.

Table 1147. CIN register field descriptions

Field	Description
0–31 PW32	32 bits of the 256-bit password challenge. Only 32-bit writes may be used.

After programming the Challenge Selector Register, the password challenge needs to be written with eight 32-bit writes from CIN0 to CIN7—in that order. The most significant 32 bits of the password challenge must be written to CIN0. Once CIN7 is written, the password is compared. The write access only completes after the comparison has been executed.

All accesses to these registers need to be made by the bus master which wrote the CHSEL register previously. If another master writes any of the CIN registers before the original master has written CIN7, the comparison fails.

68.2.1.5 Password Group *n* - Lock 0 Status register (LOCK0_PG*n*)

Each Password Group has one Lock 0 Status register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register becomes writable. For additional details of the LOCK bits, refer to the Flash memory description.

The LOCK bits do not come into effect before the life cycle has matured to *OEM Production* or older.

Offset 0x0100 LOCK0_PG0
0x0110 LOCK0_PG1
0x0120 LOCK0_PG2
0x0130 LOCK0_PG3

Access: Software
Read/Write & DCF Record Write

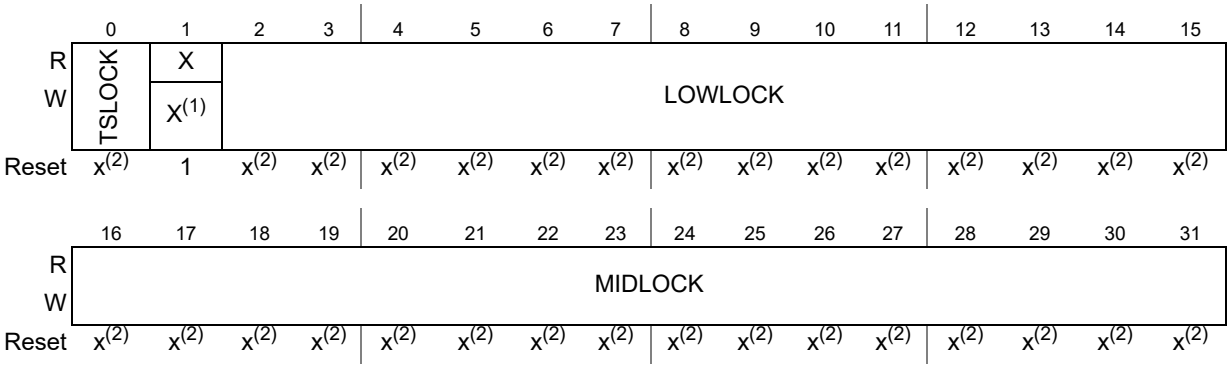


Figure 1217. LOCK0_PG*n* register

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'.

LOCK0_PG*n* register functions are as shown in *Table 1148*.

Table 1148. 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).
1	This bit is reserved and has no impact on the system.
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.

During the *Customer Delivery* life cycle, the user must program a valid password or a DCF record that writes to this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning: Failure to program valid passwords or DCF records for all password groups before the life cycle 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 are possible. However, unless the life cycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

68.2.1.6 Password Group *n* - Lock 1 status register (LOCK1_PG*n*)

Each Password Group has one Lock 1 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable. For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle matures to *OEM Production* or older.

Offset 0x0104 LOCK1_PG0
0x0114 LOCK1_PG1
0x0124 LOCK1_PG2
0x0134 LOCK1_PG3

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)

Figure 1218. LOCK1_PG*n* register

1. If no DCF records have been written to change the lock state, the bit will reset to ‘1’.

LOCK1_PG*n* register functions are as shown in *Table 1149*.

Table 1149. LOCK1_PG*n* register field descriptions

Field	Description
0–15	This bit is reserved and always has the value 0.
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.

During the *Customer Delivery* life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning: Failure to program valid passwords or DCF records for all password groups before the life cycle 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 are possible. However, unless the life cycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

68.2.1.7 Password Group *n* - Lock 2 status register (LOCK2_PG*n*)

Each Password Group has one Lock 2 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable. For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle is matured to *OEM Production* or older.

Offset 0x0108 LOCK2_PG0
0x0118 LOCK2_PG1
0x0128 LOCK2_PG2
0x0138 LOCK2_PG3

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_256LCK															
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_256LCK															
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)

Figure 1219. LOCK2_PG*n* register

1. If no DCF records have been written to change the lock state, the bit resets to '1'.

Note: Lock registers can only be accessed with 32-bit operations.

LOCK2_PG*n* register functions are as shown in *Table 1150*.

Table 1150. LOCK2_PG*n* register field descriptions

Field	Description
0–31 L_256LCK	Lower 256KByte 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.

During the *Customer Delivery* life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning: Failure to program valid passwords or DCF records for all password groups before the life cycle 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 are possible. However, unless the life cycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

68.2.1.8 Password Group *n* - Lock 3 status register (LOCK3_PG*n*)

Each Password Group has one Lock 3 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the PGL bit is cleared and the Lock registers become writable.

The LOCK bits do not come into effect before the life cycle matures to *OEM Production* or older.

Offset 0x010C LOCK3_PG0
0x011C LOCK3_PG1
0x012C LOCK3_PG2
0x013C LOCK3_PG3

Access: Software
Read/Write & DCF Record
Write

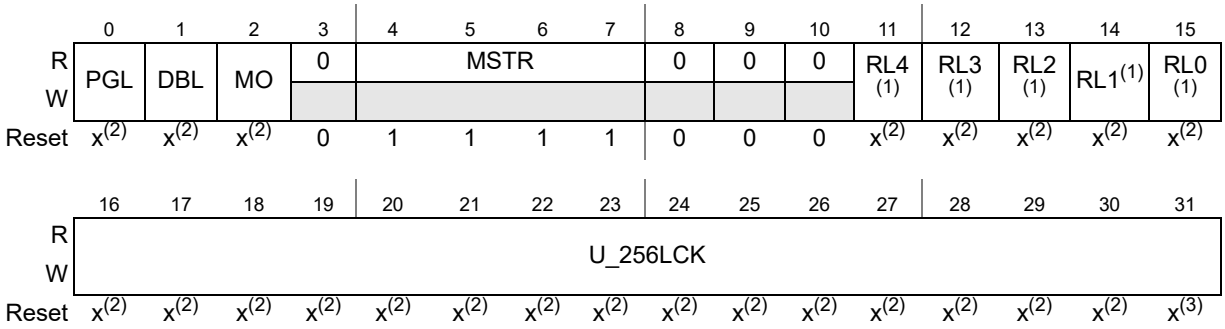


Figure 1220. LOCK3_PG*n* Register

1. Not all devices include this field. See the device reference manual's chip configuration chapter for details.
2. Reset value depends on life cycle. If life cycle 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'.

LOCK3_PG*n* register functions are as shown in *Table 1151*.

Table 1151. 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 life cycle 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
3	Reserved
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.
8–10	Reserved

Table 1151. LOCK3_PG*n* register field descriptions(Continued)

Field	Description
11 RL4	Read Lock Region 4 lock bit. Note that not all devices include this field. See the device reference manual's chip configuration chapter for details. The default value of this bit would be '1' if not updated from flash. 0Region 4 is not protected 1Region 4 is protected
12 RL3	Read Lock Region 3 lock bit. Note that not all devices include this field. See the device reference manual's chip configuration chapter for details. The default value of this bit would be '1' if not updated from flash. 0Region 3 is not protected 1Region 3 is protected
13 RL2	Read Lock Region 2 lock bit. Note that not all devices include this field. See the device reference manual's chip configuration chapter for details. The default value of this bit would be '1' if not updated from flash. 0Region 2 is not protected 1Region 2 is protected
14 RL1	Read Lock Region 1 lock bit. Note that not all devices include this field. See the device reference manual's chip configuration chapter for details. The default value of this bit would be '1' if not updated from flash. 0Region 1 is not protected 1Region 1 is protected
15 RL0	Read Lock Region 0 lock bit. Note that not all devices include this field. See the device reference manual's chip configuration chapter for details. The default value of this bit would be '1' if not updated from flash. 0Region 0 is not protected 1Region 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.

During the *Customer Delivery* life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning: Failure to program valid passwords or DCF records for all password groups before the life cycle 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 are possible. However, unless the life cycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

For operation of the region read protect feature, refer to *Section 68.4.3: Flash Memory Read Protection*.

Once a censored device enters the *Failure Analysis* life cycle, read protection is active regardless whether the system is in debug mode or not. However, it is still possible to unlock Flash regions with the correct passwords.

68.3 DCF clients

The following table shows the DCF clients for the PASS.

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 of the Reference Manual .

Table 1152. PASS DCF clients

Offset	Register	Access	Reset Value	DCF Client Address (ADDR[14:0])	Section/Page
Censorship					
None	Censorship	None	0x0000_0000	0x00B0	*68.3.1/1968*

68.3.1 Censorship

Censorship is a DCF client that is *not* accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the Flash UTEST region. The Censorship DCF client defines whether the MCU is censored or not.

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 1221. Censorship register (not accessible)

Table 1153. Censorship register field descriptions

Field	Description
0–15	Reserved.
16:31	Censorship If this value is 0x55AA the MCU is NOT censored If this value is anything other that 0x55AA, the MCU is censored

68.4 Functional description

The primary purpose of the PASS is to provide fine-grained access control to the Flash and debugger interface. It utilizes the DCF mechanism, as described in the SSCM section.

68.4.1 Censorship

Censorship is the overall control that enables and disables two independent security features: Debug Interface access; and flash memory read protection.

The MCU can be Censored by writing a DCF Record to flash that writes to the Censorship DCF client (See *Section 68.3.1, Censorship*).

68.4.2 Debug Interface Access

The debug interface in fully enabled in Customer_Delivery Lifecycle. But from OEM_Production lifecycle, it can be blocked using censorship and selectively by either entering a 256-bit password via JTAG or by software clearing the DBL bit the PASS_LOCK3 register (See *Section 68.2.1.8, 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 1154. Debug Interface Enable Truth Table

Input Conditions				Outcome
Lifecycle	**Censorship**	**Debug Enable (LOCK3[DBL] bit)**	**JTAG Password**	**Debug Interface Enable**
ST Production/ 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
	Don't Care	Don't Care	Don't Care	Blocked
Failure Analysis	Don't Care	Don't Care	Don't Care	Enabled

If the device is censored, then access is possible via JTAG password.

68.4.2.1 JTAG Password Challenge

To enable the Debug Interface Access, the JTAG password challenge registers in the JTAGC, so, it can be written by an external tool. If that password matches the JTAG password stored in UTEST flash, Debug Interface Access is enabled (according to the table above) until the next destructive reset.

68.4.3 Flash Memory Read Protection

Flash Memory Read Protection is a security feature that can selectively make regions of the flash memory unreadable. Any attempt to read a region of flash that is read protected results in the read access terminating with a transfer error. This is consistent for all bus masters including debug bus masters.

The assertion of flash memory region read protection is controlled by Lifecycle, Censorship and whether the Debug Interface access is enabled or blocked. *Table 1155 (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 (3 to 5 depending on the device). The assignment of the regions is defined in the device reference manual.

Table 1155. Flash Memory Read Protection Truth Table

INPUTS				OUTPUTS	
Lifecycle	**Censorship**	**Debug Interface Enable**	**Lock3[RLx]**	**Read Protected RL[0] (UTEST)**	**Read Protected RL[2:1] (Other Regions)**
ST Production / Customer Delivery	Don't Care	Don't Care	Don't Care	Readable	Readable
OEM Production / IN Field	Uncensored	Don't Care	Don't Care	Readable	Readable
	Censored	Blocked	Don't Care	Readable	Readable
		Enabled	Locked (1)	Read Blocked	Read Blocked
			Unlocked (0)	Readable	Readable
Failure Analysis	Uncensored	Don't Care	Don't Care	Readable	Readable
	Censored	Don't Care	Locked (1)	Readable	Read Blocked
			Unlocked (0)	Readable	Readable

68.5 Initialization/application information

68.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (see *Section 68.2.1, Register descriptions*). However, many bits receive their values based on DCF records stored in the UTEST Flash memory block.

68.5.2 Modifying Life Cycle (maturing the device)

The life cycle can be modified by an external life cycle management device like SSCM. It is no longer controlled through the PASS module.

68.5.3 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 1156. CS/ADDR Values for Lock Registers

Register	DCF CS/ADDR
PW Group 0	
Lock0	0xxxxx(1)_0100
Lock1	0xxxxx(1)_0104
Lock2	0xxxxx(1)_0108
Lock3	0xxxxx(1)_010C
PW Group 1	

Table 1156. CS/ADDR Values for Lock Registers(Continued)

Register	DCF CS/ADDR
Lock0	0xxxxx(1)_0110
Lock1	0xxxxx(1)_0114
Lock2	0xxxxx(1)_0118
Lock3	0xxxxx(1)_011C
PW Group 2	
Lock0	0xxxxx(1)_0120
Lock1	0xxxxx(1)_0124
Lock2	0xxxxx(1)_0128
Lock3	0xxxxx(1)_012C
PW Group3	
Lock0	0xxxxx(1)_0130
Lock1	0xxxxx(1)_0134
Lock2	0xxxxx(1)_0138
Lock3	0xxxxx(1)_013C

1. Value depends on the DCF Client Select assigned to the PASS module in the device. See Device reference manual for the Client Select assignment

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 1157*.

Table 1157. Setting some bits in the MIDLOCK field

ADDR offset	DATA	Description
0x00	0x05AA_55AF	Start Record
0x04	0x0000_0000	
0x08	0x0000_00FF	PW Group 1, Lock0 Set MIDLOCK to 0xFF
0x0C	0xxxxx(1)_0110	
0x10	0xFFFF_FFFF	Stop Record
0x14	0xFFFF_FFFF	
...	...	

1. Value depends on the DCF Client Select assigned to the PASS module in the device. See Device reference manual for the Client Select assignment

68.5.4 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 is defined in the Device Reference Manual) and the address 0x00B0. Therefore 0xxxxx_00B0. 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 uncensored.

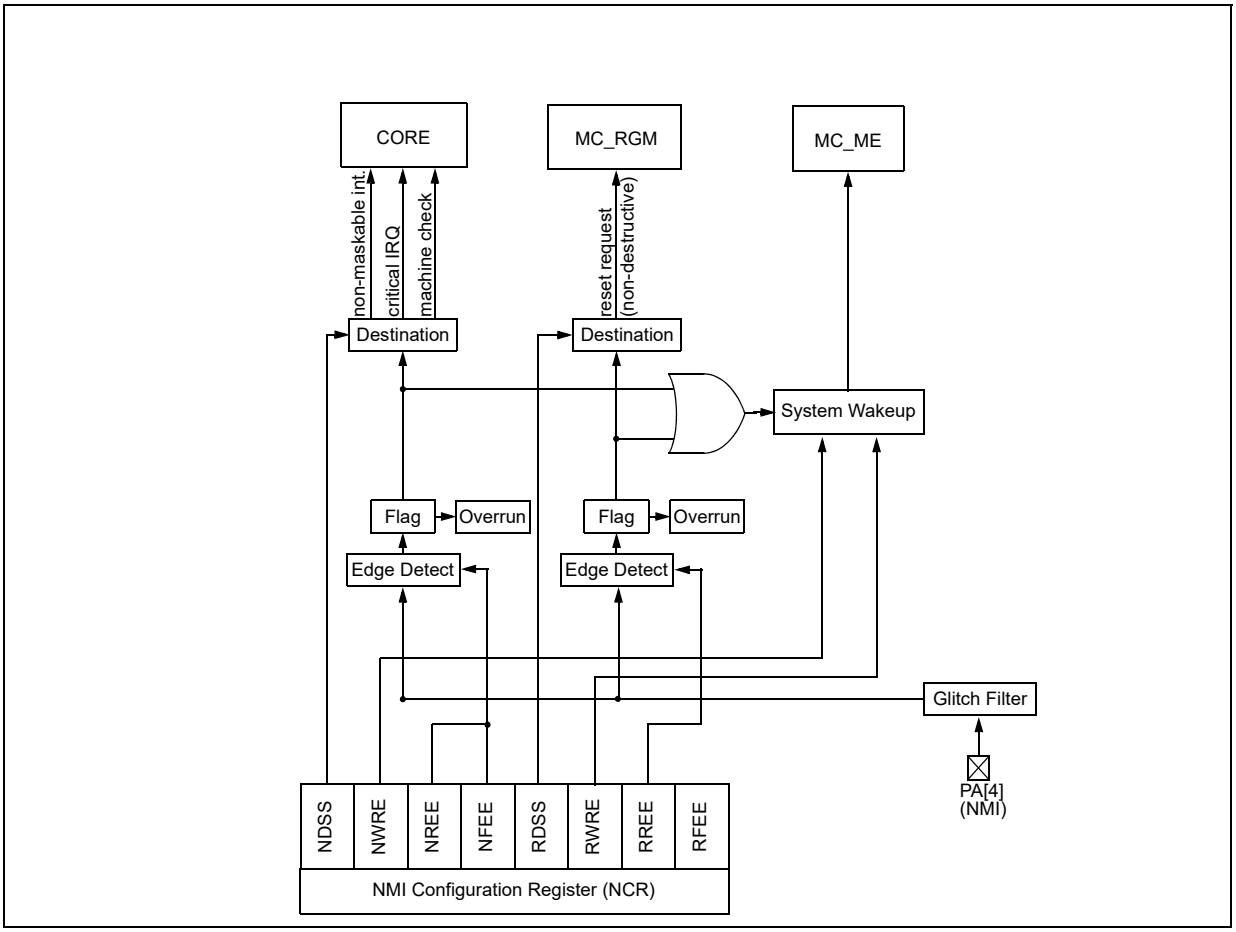
69 Wakeup Unit (WKPU)

69.1 Introduction

69.1.1 Overview

The Wakeup Unit (WKPU) supports one external source that can cause non-maskable interrupts to the on-chip core and wakeup events to the system. This NMI external source can also generate a functional reset event to the MC_RGM. *Figure 1222* is a block diagram of the WKPU and its interfaces to other system components.

Figure 1222. WKPU block diagram



69.1.2 Features

The WKPU supports these distinctive features:

- one external NMI source
- one analog glitch filter
- independent interrupt destination for the core:
  - non-maskable interrupt
  - critical interrupt
  - machine check request
- 13 I/O external interrupts: 4–14, 20, and 21 wakeup lines
- active edge selection control for events to the core and functional reset event to MC_RGM
- configurable wakeup event from low power mode to the MC_ME
- configurable wakeup event to exit from STOP mode to the MC_ME
- WKPU register configuration lock

69.2 External signal description

Active-edge-selectable, alternate-function pin PA[4] triggers the wakeup unit. See “Signal Description” chapter for further pin details.

69.3 Memory map and register description

This section provides a detailed description of all registers accessible in the WKPU module.

69.3.1 Memory map

Table 1158 gives an overview of the WKPU registers.

Table 1158. WKPU memory map

Address offset	Register	Size (bits)	Supported access sizes (bits)	Location
0x0000	NMI Status Flag Register (NSR)	32	32/16/8	*on page 1976*
0x0004	Reserved			
0x0008	NMI Configuration Register (NCR)	32	32/16/8	*on page 1977*
0x0014	Wakeup/Interrupt Status Flag Register (WISR)	32	32	*on page 1979*
0x0018	Interrupt Request Enable Register (IRER)	32	32	*on page 1980*
0x001C	Wakeup Request Enable Register (WRER)	32	32	*on page 1980*
0x0020 – 0x0027	Reserved			
0x0028	Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)	32	32	*on page 1981*

Table 1158. WKPU memory map(Continued)

Address offset	Register	Size (bits)	Supported access sizes (bits)	Location
0x002C	Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)	32	32	on page 1981
0x0030 – 0x3FFF	Reserved			

Note: Reserved registers will read as 0, writes will have no effect. If supported and enabled by the SoC, a transfer error will be issued when trying to access completely reserved register space.

69.3.2 Register descriptions

This section describes in address order all the WKPU 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.

69.3.2.1 NMI Status Flag Register (NSR)

This register holds the status flags of the NMI external source pin and reset request. 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 or reset request event is detected. The overrun flag is set whenever an NMI or reset request event is detected and the status flag is set (i.e. has not yet been cleared). The reset request overrun is a don't-care since the device will be reset on the first event.

The status flags are set independently of the NDSS/RDSS bits. Therefore, these flags can set but not generate a non-maskable interrupt or system wakeup if NWRE/RWRE = 0 or if NDSS is set to "no request".

Note: The overrun flag is cleared by writing a '1' to the appropriate overrun bit in the NSR. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

Note: NIF is set when: NMI Destination Source Select is disabled (NDSS = 11) and NMI System Wakeup Request is disabled (NWRE = 0) and an NMI input event is seen (pad4 is toggled).

Note: Also see notes in Section 69.3.2.2, NMI Configuration Register (NCR).

Address: 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
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIF	NOVF	0	0	0	0	0	0	RIF	ROVF	0	0	0	0	0	0
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1223. NMI Status Flag Register (NSR)

Table 1159. NSR field descriptions

Field	Description
NIF	NMI Status Flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE or NFEE set), an NMI input event will set NIF. NIF causes an interrupt request based on the NMI destination, as configured by the NDSS bit, and/or a system wakeup request based on the NWRE bit configuration. 1 An event as defined by NREE and NFEE has occurred 0 No event has occurred on the pad
NOVF	NMI 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 NIF value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. NOVF causes an interrupt request based on the NMI destination, as configure by the NDSS bit, and/or a system wakeup request based on the NWRE bit configuration. 1 An overrun has occurred on NMI input 0 No overrun has occurred on NMI input
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 ESR1 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
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 ESR1 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

69.3.2.2 NMI Configuration Register (NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

Address: 0x0008 Access: User read/write

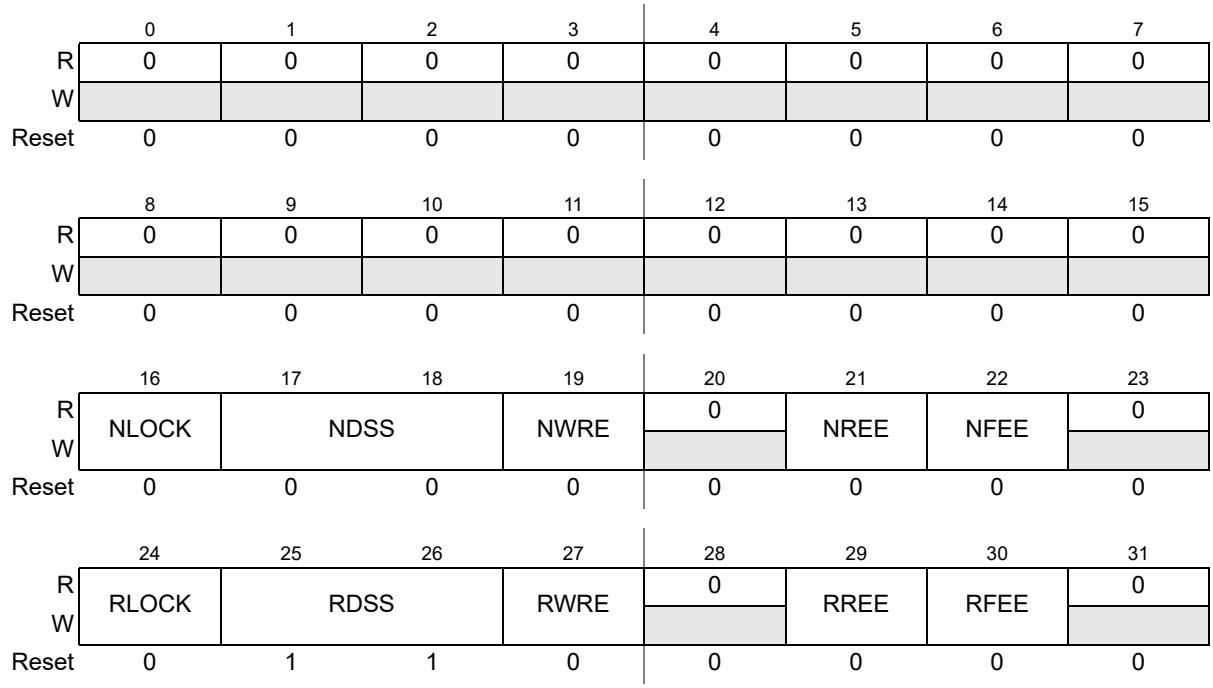


Figure 1224. NMI Configuration Register (NCR)

Table 1160. NCR field descriptions

Field	Description
NLOCK	NMI Configuration Lock Register 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.
NDSS	NMI Destination Source Select 00 Non-maskable interrupt. A non-maskable interrupt will occur if NIF or NOVF is set. 01 Critical interrupt. A critical interrupt will occur if NIF or NOVF is set. 10 Machine check request. A machine check request will occur if NIF or NOVF is set. 11 No reaction
NWRE	NMI System Wakeup Request Enable 1 A set NIF bit or set NOVF bit causes a system wakeup request 0 System wakeup requests from the corresponding NIF bit are disabled
NREE	NMI Rising-edge Events Enable 1 Rising-edge event is enabled 0 Rising-edge event is disabled
NFEE	NMI Falling-edge Events Enable 1 Falling-edge event is enabled 0 Falling-edge event is disabled
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.

Table 1160. NCR field descriptions(Continued)

Field	Description
RDSS	Reset Destination Source Select 00 Reset request to RGM 01 No reaction 10 No reaction 11 No reaction
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
RREE	Reset Rising-edge Events Enable 1 Rising-edge event is enabled 0 Rising-edge event is disabled
RFEE	Reset Falling-edge Events Enable 1 Falling-edge event is enabled 0 Falling-edge event is disabled

Note: Writing a '0' to both NREE[x] and NFEE[x] disables the NMI functionality completely (i.e., no system wakeup or interrupt is generated on any pad activity).

Note: Writing to the NCR when NIF = 1 or NOVF = 1 causes an interrupt to the destination source/system wakeup to be generated independent of an NMI wakeup event.

69.3.2.3 Wakeup/Interrupt Status Flag Register (WISR)

This register holds the wakeup/interrupt flags.

Address: 0x0014 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	0	0	0	0	0	0	0	0	0	0	0	0	0	EIF[18]	EIF[17]	0	0	EIF[14:4]											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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1225. Wakeup/Interrupt Status Flag Register (WISR)

Table 1161. WISR field descriptions

Field	Description
EIF[x]	External Wakeup/Interrupt Status Flag x. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER[x]), EIF[x] causes an interrupt request. 1An event as defined by WIREER and WIFEER has occurred 0 No event has occurred on the pad

Note: Status bits associated with on-chip wakeup sources are located to the left of the external wakeup/interrupt status bits and are read only. The wakeup for these sources must be configured and cleared at the on-chip wakeup source. Also, the configuration registers for the external interrupts/wakeups do not have corresponding bits.

Note: The external interrupts are listed in the "I/O signal description and input multiplexing" table which is contained in a Microsoft Excel® workbook attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

69.3.2.4 Interrupt Request Enable Register (IRER)

This register is used to enable the interrupt messaging from the wakeup/interrupt pads to the interrupt controller.

Address: 0x0018 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	EIRE[18]	EIRE[17]	0	0	EIRE[14:4]											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 1226. Interrupt Request Enable Register (IRER)

Table 1162. IRER field descriptions

Field	Description
EIRE[x]	External Interrupt Request Enable x. 1 A set EIF[x] bit causes an interrupt request 0 Interrupt requests from the corresponding EIF[x] bit are disabled

Note: The external interrupts are listed in the "I/O signal description and input multiplexing" table which is contained in a Microsoft Excel® workbook attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

69.3.2.5 Wakeup Request Enable Register (WRER)

This register is used to enable the system wakeup messaging from the wakeup/interrupt pads to the mode entry and power control modules.

Address: 0x001C 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	WRE[18]	WRE[17]	0	0	WRE[14:4]											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 1227. Wakeup Request Enable Register (WRER)

Table 1163. WRER field descriptions

Field	Description
WRE[x]	External Wakeup Request Enable x. 1 A set EIF[x] bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF[x] bit are disabled

Note: The external interrupts are listed in the "I/O signal description and input multiplexing" table which is contained in a Microsoft Excel® workbook attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

69.3.2.6 Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)

This register is used to enable rising-edge triggered events on the corresponding wakeup/interrupt pads.

Address: 0x0028 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	IREE[18]	IREE[17]	0	0	IREE[14:4]											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 1228. Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)

Table 1164. WIREER field descriptions

Field	Description
IREE[x]	External Interrupt Rising-edge Events Enable x. 1 Rising-edge event is enabled 0 Rising-edge event is disabled

Note: The external interrupts are listed in the "I/O signal description and input multiplexing" table which is contained in a Microsoft Excel® workbook attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

69.3.2.7 Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)

This register is used to enable falling-edge triggered events on the corresponding wakeup/interrupt pads.

Address: 0x002C 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	IFEE[18]	IFEE[17]	0	0	IFEE[14:4]											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 1229. Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)

Table 1165. WIFEER field descriptions

Field	Description
IFEEx	External Interrupt Falling-edge Events Enable x. 1 Falling-edge event is enabled 0 Falling-edge event is disabled

Note: The external interrupts are listed in the "I/O signal description and input multiplexing" table which is contained in a Microsoft Excel® workbook attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

69.4 Functional description

This section provides a complete functional description of the WKPU features.

69.4.1 Non-maskable interrupts

The WKPU allows the external NMI pin (ESR1) to assert the non-maskable interrupts of the cores on the device. These interrupts include non-maskable interrupts, critical interrupts, or machine check requests to the core.

The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits. See *Table 1160* for details.

The enabling of rising edge, falling edge, or either edge reaction to the NMI pin is independent for the core, and is controlled by NFEE[2:0] and NREE[2:0] bits in the NCR.

The WKPU supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.

The NMI supports a status flag and an overrun flag, which are located in the NSR. See *Section 69.3.2.1, NMI Status Flag Register (NSR)* for more information.

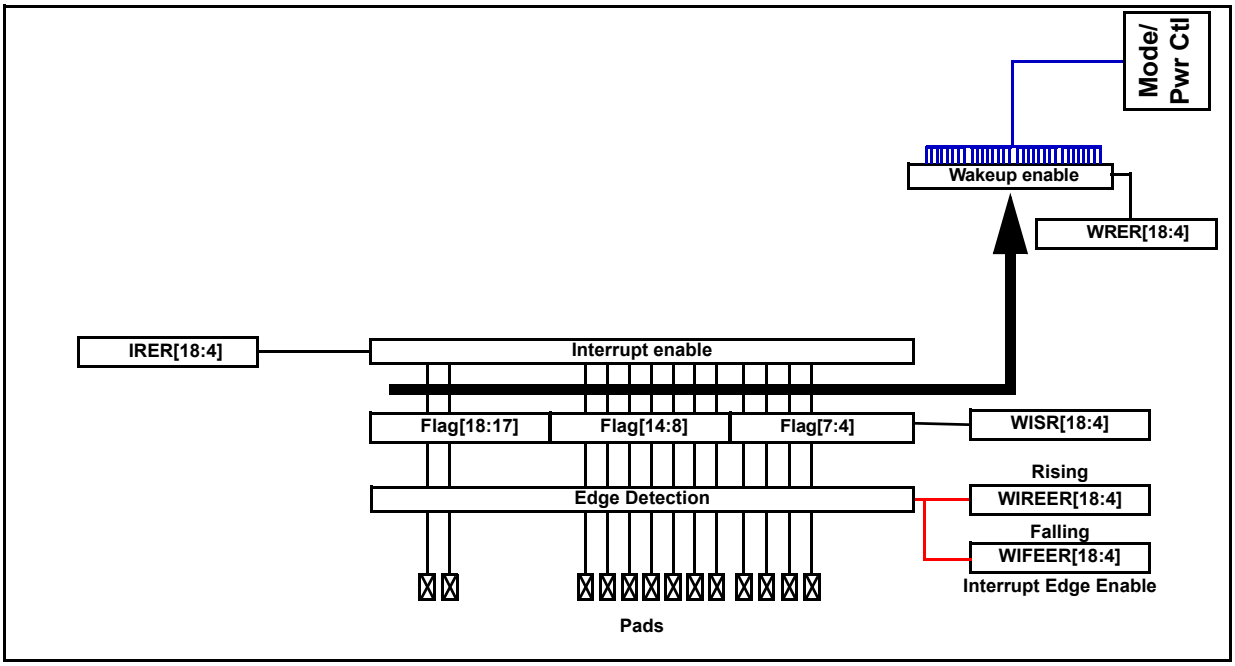
69.4.2 External Wakeups/Interrupts

The WKPU supports 13 external wakeup/interrupts which can be allocated to any pad necessary at the SoC level. This allocation is fixed per SoC.

The WKPU supports 3 interrupt vectors to the interrupt controller of the SoC. Each interrupt vector can support up to the number of external interrupt sources from the device pads with the total across all vectors being equal to the number of external interrupt sources. Each external interrupt sources is assigned to exactly one interrupt vector. The interrupt vector assignment is sequential so that one interrupt vector is for external interrupt sources 0 through N-1, the next is for N through N+M-1, and so forth.

Refer to *Figure 1230* for an overview of the external interrupt implementation.

Figure 1230. External Interrupt Pad diagram



All of the external interrupt pads within a single group 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 SoC.

69.4.2.1 External Interrupt Management

Each interrupt can be enabled or disabled independently. This can be performed using a single rolled up register (*Figure 1226*). 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.

Note: Writing a '0' to both IREE[x] and IFEE[x] disables the external interrupt functionality for that pad completely (i.e., no system wakeup or interrupt will be generated on any activity on that pad).

The active IRQ edge is controlled by the users through the configuration of the registers WIREER and WIFEER.

Each external interrupt supports an individual flag which is held in the flag register (WISR). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

69.4.3 Reset management

The WKPU allows the external NMI pin (ESR1) to assert a non-destructive reset request to the MC_RGM block. The ESR1 reset request can be reconfigured in the MC_RGM to generate a SAFE mode or interrupt request (see *Chapter 51: Reset Generation Module (MC_RGM)* for details).

The enabling of rising edge, falling edge, or either-edge reaction to the NMI pin is independent for the reset request, and is controlled by the RFEE/RREE bits in the NCR.

Reset request supports a status flag and an overrun flag, which are located in the NSR. See *Section 69.3.2.1, NMI Status Flag Register (NSR)* for more information.

69.4.4 System wakeup

The WKPU allows the external NMI pin (ESR1) to assert a system wakeup from STOP to the MC_ME block on the device. The system wakeup enable is asserted in the NCR by any of the NWRE/RWRE bits.

69.4.5 Glitch filter

An analog glitch filter is placed between the NMI pin (ESR1) and the WKPU block. The filter is enabled by the NFE[0] bit in the NCR. See the device data sheet for the glitch filter electrical specifications.

69.4.5.1 Edge detect

Each NMI and reset request can be enabled or disabled independently. This can be performed using the single NCR laid out to contain all configuration bits for a given NMI in a single byte (see *Figure 1224*). The user can configure interrupts with an active rising edge, an active falling edge, or either edge being active. Disabling both edge events results in no interrupt or reset being propagated from the WKPU block.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits. The active reset request edge is controlled by the RREE and RFEE bits.

Note: After reset, NREE/RREE and NFEE/RFEE are set to '0', therefore the NMI functionality and reset request is disabled after reset and must be enabled explicitly by software.

69.5 Initialization information

This section discusses initialization for the following features:

- glitch filter and pad configuration
- non-maskable interrupt initialization
- reset request

69.5.1 Glitch filter and pad configuration

Glitch filter control and pad configuration should be performed while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

Note: When enabling the glitch filter, do not enable the rising-/falling-edge events bits (NREE[x], NFEE[x], RREE, RFEE) in the same register write.

69.5.2 Non-maskable interrupt initialization

When an NMI 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:

1. mask interrupts by clearing the NIF[n] and NOVF[n] bits in WKPU_NSR
2. clear WKPU_NCR bit NFEE[n] and NREE[n], preferably by writing 0 to all bits in this register
3. enable the analog glitch filter as desired by writing the NFE[0] bit in the WKPU_NCR
4. configure the appropriate SIUL2_MSCR for the desired NMI interrupt pin(s) as follows:
   – clear the ODC bits to disable output
   – set the IBE bit to enable the pin's input buffer
   – if the internal weak pull-up/pull-down is used, configure the appropriate WPUE and WPDE bits

Note: NMI interrupt pins should never be configured as outputs (MSCR_ODC bits are not zeros) when external interrupt inputs are desired since false interrupts could be detected (such as from a GPIO configuration).

5. write appropriate bit values to the WKPU_NCR to configure:
   – NMI source (NDSS[n])
   – wakeup request (NWRE[n])
   – glitch filter (NFE[0])
   – rising/falling edge events enable (NFEE[n] and NREE[n])
   – NMI configuration lock (NLOCK[n])

69.5.3 Reset request

When an NMI reset request pin is first enabled, it is possible to get a false reset request. To prevent a false reset request during pin reset request initialization the user must do the following:

1. mask reset requests by clearing the RIF and ROVF bits in WKPU_NSR
2. clear WKPU_NCR bit RFEE and RREE, preferably by writing 0 to all bits in this register
3. enable the analog glitch filter as desired by writing the NFE[0] bit in the WKPU_NCR
4. configure the appropriate SIUL2_MSCR for the NMI reset request pin(s) as follows:
   – clear the ODC bits to disable output
   – set the IBE bit to enable the pin's input buffer
   – if the internal weak pull-up/pull-down is used, configure the appropriate WPUE and WPDE bits

Note: NMI reset request pins should never be configured as outputs (MSCR_ODC bits are not zeros) when external reset request inputs are desired since false reset requests could be detected (such as from a GPIO configuration).

5. write appropriate bit values to the WKPU_NCR to configure:
   – reset destination source (RDSS])
   – reset wakeup request (RWRE)
   – glitch filter (NFE[0])
   – rising/falling edge events enable (RFEE and RREE)
   – reset configuration lock (RLOCK)

Appendix A Acronyms and abbreviations

Table 1166 lists acronyms and abbreviations used in this document.

Table 1166. Acronyms and abbreviations

Term	Meaning
ATX	Analog test module
APU	– Advanced protocol unit – Auxiliary processing unit
AUTOSAR®	Automotive Open System Architecture
BAF	Boot assist flash
BIST	Built-in self test
BIU	Bus Interface Unit
CAN	Controller area network
CCCU	Clock calibration on CAN unit
CF	Critical fault
CGM	Clock generation module
CMU	Clock monitoring unit
CPU	Central processing unit
CQM	Clock quality monitor
CRC	Cyclic redundancy check
LFAST	Interprocessor communication module
DMA	Direct memory access
DMAMUX	Direct memory access controller multiplexer
DSPI	Deserial serial peripheral interface
DTS	Development tool semaphore
ECC	Error correction code
EMI	Electromagnetic interference
EOUT	Error out
ESD	Electromagnetic static discharge
FCCU	Fault control and collection unit
FEC	Fast Ethernet controller
FLEXRAY	Automotive network communications protocol
FMPLL	Frequency-modulated phase-locked loop
FOSU	FCCU output supervision unit
FPEC	Flash program erase controller
FSM	Finite state machine

Table 1166. Acronyms and abbreviations(Continued)

Term	Meaning
GPIO	General-purpose I/O
GTM	Generic timer module
HW	Hardware
ID	LIN identifier field
IEEE	Institute for Electrical and Electronics Engineers
IIC	Inter-integrated circuit (I^2C)
IMA	Indirect memory access
INTC	Interrupt controller
INTF	Interface
IP	Intellectual propriety
IPS	– Internal peripheral system – IPI-SkyBlue slave bus
IRQ	Interrupt request
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
MBIST	Memory built-in self test
MC	MC_CGM, MC_ME, MC_PCU, and MC_RGM modules
ME	Mode entry module (MC_ME)
MEMU	Memory error management unit
MFD	Multiplication factor divider
Mux	Multiplex
NCF	Non-critical fault
NMI	Non-maskable interrupts
NVM	Nonvolatile memory
OSC	Oscillator
OTP	One time programmable
PASS	Password and device security module
PCM	Platform configuration module
PCU	Power control unit
PFLASH	Flash controller
PIT	Periodic interrupt timer

Table 1166. Acronyms and abbreviations(Continued)

Term	Meaning
PLL	Phase lock loop
PMC	Power management controller
PRAM	RAM controller
PSI5	Peripheral sensor interface
PWD	Password
RCC	Redundancy control checkers
RCCU	Redundancy control checker unit
RCOSC	RC oscillator
RF	Recoverable fault
RFD	Reduced frequency divider
RGM	Reset generation module
RTL	Register transfer language
RWW	Read-while-write (operation)
Rx	Receive
SAG	Safety application guide
SAR ADC	Successive approximation register analog-to-digital converter
SD ADC	Sigma-delta analog-to-digital converter
SEMA4	Semaphore
SENT	Single edge nibble transmission
SIL	Safety integrity level
SIPI	System interprocessor interface
SIUL	System integration unit lite
SMPU	System memory protection unit
SoC	System-on-chip
SOR	Sphere of redundancy
SSCM	System status and configuration module
STM	System timer module
SW	Software
SWT	Software watchdog timer
TBD	To be defined
TCD	Transfer control descriptor
TSENS	Temperature sensor
TTCAN	Time triggered CAN
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter

Table 1166. Acronyms and abbreviations(Continued)

Term	Meaning
UF	Unrecoverable fault
VCO	Voltage controlled oscillator
WDG	Watchdog timer
WKPU	Wakeup unit
WS	Wait state
XBAR	Crossbar

Revision history

Table 1167. Revision history

Date	Rev	Changes
18-Jul-2012	1	Initial release
03-Apr-2013	2	Throughout document: Formatting and editorial changes; removed references to the Tamper Detection module; “UTest” updated to “UTEST”; Removed residual references to FCCU, HSM and Buddy/Emulation Device. *Chapter 1: Preface*: no substantial changes *Chapter 2: Introduction* *Section 2.2, Features summary*: – Updated PIT features – Updated CPU features *Figure 2 (Block diagram)*: changed “AIPS Bridge” to “Peripheral Bridge” *Figure 3 (Periphery allocation)*: renamed several modules; removed “BAM” module *Section 2.5, Software debug and calibration*: removed *Performance monitors for CPU* from features list *Chapter 3: Embedded memories* *Figure 5 (McKinley flash memory block diagram)*: updated to show cache as 4-entry, 2-way *Figure 3.3.1 (Flash memory controller)*: corrected content to state device has 4-entry, 2-way cache Removed section “Selecting flash blocks” *4: Signal description* *Table 6: Pin startup and reset states*: removed F0 (ERROR0) row *Table 9: LVDSM pin descriptions*: added “Interprocessor Bus LFAST, LVDS Transmit Positive Terminal” row *5: Memory map* *Table 11: Peripheral (PBRIDGE_A) memory map*: updated selected names in description column; added column “PBRIDGE Access Control Register” *Table 12: Flash memory and overlay RAM map*: removed footnote “Cut 1.0: implementation is 8 x 16 KB but...” *Table 14: UTEST flash memory map*: updated “Comments” cell for Unique Identifier (UID) *6: Device configuration* Changed all instances of “MCAN” to “M_CAN“; changed all instances of “TTCAN” to “M_TTCAN”; changed all instances of all instances of PLL_DIG changed to PLLDIG; changed all instances of all instances of OSC_DIG changed to XOSC *Figure 15 (Crossbar switch integration)*: – M0 connects to “Core data bus” (was Core instruction bus) – M1 connects to “Core instruction bus” (was Core data bus)

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	*Table 15: Reset settings for e200z215An3 resources*: – Removed one instance of footnote "Reset by processor reset... unconditionally by POR" (previously listed twice) – Removed rows "L1CFG0", "L1CFG1", "L1CSR0, 1" and "L1FINV0, 1" *Table 18: Crossbar switch master assignments*: – Changed master module description for Port 0 to "Core data bus" (was core instruction bus) – Changed master module description for Port 1 to "Core instruction bus" (was core data bus) *Section 6.3.5, Platform Configuration Module (PCM)*: Removed references and content relating to FEC BURST Optimization Control Register (FBOMCR) and IAHB Burst Enable 1(IAHB_BE1) register *Table 23: Interrupt sources*: – Changed Source Descriptions and Source names from "IRQ#837" to "IRQ#848" – Updated source description and source name for IRQ 484 (was ipi_int_ext_pd1_done / ipi_int_ext_pd1_done; is PLL_0_1 / PLL0SR[EXTPDF]) Table 33 (ERM memory assignment): updated "Assigned memory" descriptions for rows "Memory0, Memory4 and Memory5" Table 34 (EIM memory assignment): updated "Assigned memory" descriptions for rows "Memory4 and Memory5" Section 6.6, Timers: changed number of generic 32-bit timer channels for PIT module to four (was eight) *Table 36: Timers*: changed "GTM101 Integration Module Configuration" to "GTM101 Generic Timer Module" Updated introduction of Section 6.7.2.3, Default configuration *Table 41: PIT0 memory map*: changed range of Timer Channels to [0:3] (was [0:7]) Removed section "Fast Ethernet Controller (FEC) configuration" *Section 6.7.1, FEC interface*: renamed (was FEC interface selection); updated content. Removed section CAN subsystem configuration *Section 6.7.3.2, Device specific features*: changed SIPI CHIP ID to "0AF0201Dh0AF02041h" was "0AF0601Dh0AF06041h" Added *Section 6.7.4, DSPI configuration* *Figure 24 (DSPI Module Configuration Register (DSPIx_MCR))*: – Changed bit 16 to reserved – Changed bit 28 to "XSPI" Added *Table 48: DSPIx_MCR field descriptions* Added Section 6.7.7, LINFlexD implemented registers *Section 6.7.12.2, Clock Monitor Unit (CMUIOP) protected registers*: renamed (was I/O Processor Clock Monitor Unit (CMUIOP) protected registers) *Table 50: LINFlexD configurations*: changed the number of filters for linflex_1 and linflex_14 from '16' to '0' Section 6.3.8.1, DMA channel assignment: updated number of multiplexers to 2 (was 4)

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	*Chapter 8: Reset and Boot* Removed all instances of "[FUNC]" and "[DEST]" *Section 8.3.2, Power-up phase: power stabilization*: removed following sections: – PHASE1[FUNC] Phase: temporization and monitoring setup – PHASE2[FUNC] Phase: flash initial configuration – PHASE3[FUNC] Phase: device configuration monitoring *Section 8.3.3, PHASE0 Phase: analog supply initial configuration*: correction—flash memory is not initialized during PHASE0 Section 7.4.3.2, Reset Generation Module has entered idle state (state 1 and 2): removed "LBIST and MBIST have completed their test cycles" Table 65 (Module status during reset phases): removed "P1[DEST]-P3[DEST]" columns Section 7.4.4.7, Core continues BAF code execution (state 23): changed "IOP" to "Core" in title *Section 8.4.6.9, IOP begins BAF code execution (state 22)*: changed "IOP" to "Core" in title *Chapter 9: Device Configuration Format (DCF) Records* Removed references to STCU2, tamper detection, LBIST and MBIST *Table 112: DCF client list*: – Added OTP section with 4 corresponding OTP_ENx registers – Set UTEST Miscellaneous reset value to 0x00000941 *Figure 45 (UTEST Miscellaneous DCF Client)* and *Table 114: UTEST Miscellaneous DCF Client field descriptions*: set bit 18 to "reserved" (was 3p3V_5V) Updated Figure 33 (PMC_REE bits DCF Client): Updated Table 74 (PMC_REE bits DCF client field descriptions) *Chapter 10: Power management* Major content revision to reflect new voltage detector setup *Figure 49 (Device high voltage power-ground network)*: major update *Section 10.2.1, Low power support and STOP mode implementation*: updated content describing how peripheral modules can be enabled/disabled in STOP mode; modified quantities referring to CAN modules. Table 71 (SYSTEM modes): deleted "STANDBY wakeup" from DRUN mode Entry column Table 72 (USER modes): deleted "STANDBY through S/W" from RUNx mode Exit column *Table 117: Device power management controller external signals*: removed "VDD_LV_BD" Table 74 (POR and voltage monitors description): added definitions of REE and IE to footnote Table 75 (Voltage monitors configurability): replaced "IEE" with "IE" in header row; added footnotes to define REE, RES, and IE *Table 122: Functional terminal state during power-up and reset*: changed TMS POWERUP pad state from "high impedance" to "weak pull-up"; changed TDI POWERUP pad state from "high impedance" to "weak pull-up"; changed TDO RESET pad state from to "weak pull-up" to "high impedance".

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	*Chapter 11: Security* *Section 11.1, Basic security*: removed "BAR software routines to initialize part..." from security features list *11: Calibration and Debug* *Table 112: DCF client list*: – Added OTP section with 4 corresponding OTP_ENx registers – Set UTEST Miscellaneous reset value to 0x00000941 *Section 11.2.1, Debug and Calibration Interface (DCI)*: removed "Port sharing logic..." Removed section "Compact JTAG (CJTAG)" *Table 82: DCI_PINCR register field descriptions*: bits 3–6 and 11–15 set to "reserved" *Section 11.2.3.1, SPU Level1 input Mux configurations registers*: removed 26 MUX inputs MUX 22 [110]–MUX 24 [110], MUX 34 [110], MUX 35 [110], MUX 36 [000], MUX 37 [100], MUX 37 [000], MUX 38 [000], MUX 38 [010], MUX 39 [000], MUX 48 [011]–MUX 55 [011], MUX 56 [010]– MUX 62 [010] *Table 85: L1SEL2 register field descriptions*: removed MUX inputs MUX 20 [110] and MUX 21 [110] *Table 87: L1SEL4 register field descriptions*: removed MUX inputs MUX 34 [110], MUX 35 [110], MUX 36 [000]–MUX 38 [000], MUX 38 [010], and MUX 39 [000] *Section 11.5.2, Nexus Aurora Router (NAR)*: end address for overlay RAM changed to 0x0D001FFF *12: Core e200z215An3 description*: no substantial changes *Chapter 16: System Integration Unit Lite2 (SIUL2)* Updated SIUL2 MSCR names to I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511) and Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR_MUX_512–SIUL2_MSCR_MUX_1023) *Section 16.1.2, Features*: – Updated number of GPIO ports (is 7, was 1 to 32) – Updated number of interrupt sources (is 3, was 1 to 32) *Table 199: SIUL2 memory map*: row 0x0040–0x00BF end address changed to 0x00BC; 0x0240–0x123F end address changed to 0x123C; row "SUIL2 User Defined Register 0–User Defined Register 63 (SIUL2_UDR0–SIUL2_UDR63)" changed to Reserved *Table 200: SIUL2_MIDR1 field descriptions*: updated description of field "PKG" *Table 201: SIUL2_MIDR2 field description*: updated description of field "FAMILYNUM" *Section 16.2.2.3, SIUL2 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0)*: added fields relative to bits 28–30 *Section 16.2.2.4, SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)*: added fields relative to bits 28–30 *Section 16.2.2.5, SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)*: added fields relative to bits 28–30 *Section 16.2.2.6, SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)*: added fields relative to bits 28–30

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	*Section 16.2.2.7, SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)*: added fields relative to bits 28–30 *Section 16.2.2.8, SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)*:added fields relative to bits 28–30 Removed section "User Defined Register (SIUL_UDR0 – SIUL_UDR63)" *Figure 113 (MSCR I/O pin and IP block port connectivity)*: removed IP Block n; renamed original "IP Block m" to "IP Block n" *Figure 114 (I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511))*: modified OCR field to encompass 3 bits from 5 to 7 (was 6:7) *Table 211: I/O MSCR reset state exceptions*: – Removed rows for PB[11], PI[14:15], PK[12:15], and PM[0:8] ports – Added "SSS" column Added *Table 210: SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511 field description* *Table 212: SIUL2_MSCR_MUX_512–SIUL2_MSCR_MUX_1023 field description*: removed all fields except for "INV" and "SSS" *Section 16.2.2.11.4, Module-port MSCR assignments and SSS values*: removed Table (Peripheral and input multiplexing assignments) *Figure 116 (SIUL2 GPIO Pad Data Output registers (SIUL2_GPDO0–SIUL2_GPDO511))*: reorgnanized bitmap (was 32-bit register with four PDO fields (0-3); is 8-bit register with single PDOx field) *Figure 117 (SIUL2 GPIO Pad Data Input Registers (SIUL2_GPDI0–SIUL2_GPDI511))*: reorgnanized bitmap (was 32-bit register with four PDI fields (0-3); is 8-bit register with single PDIx field) *Chapter 17: Crossbar Switch (XBAR)* *Table 245: XBAR memory map*: removed 10 rows XBAR0_PRS3–XBAR0_PRS7 and XBAR0_CRS3–XBAR0_CRS7 *Figure 143 (Priority Registers Slave (XBAR_PRSn))*: removed addresses FC00_4300h (XBAR0_PRS3), FC00_4400h (XBAR0_PRS4), FC00_4500h (XBAR0_PRS5), FC00_4600h (XBAR0_PRS6), and FC00_4700h (XBAR0_PRS7); removed field names "M3" to "M7" *Table 246: XBAR_PRSn field descriptions*: changed field description for bits 0–20 to "Reserved"; removed field descriptions "M3" to "M7" *Figure 144 (XBAR Control Register (XBAR_CRSn))*: removed addresses FC00_4310h (XBAR0_CRS3), FC00_4410h (XBAR0_CRS4), FC00_4510h (XBAR0_CRS5), FC00_4610h (XBAR0_CRS6), and FC00_4710h (XBAR0_CRS7); changed HPE field bit range to 13–15 (was 8–15); bits 8–12 changed to "Reserved" *Table 247: XBAR_CRSn field descriptions*: changed HPE bit range to 13–15; set bit range 2–12 to "Reserved" *Chapter 20: Peripheral Bridge* *Table 259: Peripheral bridge memory map*: register at address offsets 0x004 and 0x110 changed to "Reserved" (were AIPS_MPRB and AIPS_PACRE) *Section 20.2.1, Master Privilege Registers (AIPS_MPRA & AIPS_MPRB)*: removed "AIPS_MPRB" from section title

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	Removed figure "Master Privilege Register B (MPRB)" *Table 261: PACR memory map*: register at row 0x110 changed to "Reserved" (was AIPS_PACRE) *Chapter 7: System Memory Protection Unit (SMPU)* *Table 130: SMPUx memory map*: added footnote about SMPU0_RGDn_WORD2_FMT0 and SMPU0_RGDn_WORD2_FMT1 implementation Chapter 17, Intelligent AHB Gasket (IAHBG): no substantial changes Chapter 18, Interrupt Controller (INTC): Removed all references to the INTC_MPROT register, its fields, and related functionality In the register descriptions, redefined the SSCIRs to be 8 bits (instead of 32 bits) in size *Section 1.1, Introduction*: changed "and automotive powertrain applications" to "and is capable of processing high-demand interrupt sources" *Section 1.6.1.3, Unique vector for each interrupt request source*, rewrote the last sentence to include the correct starting peripheral interrupt request number Revised *Section 1.6.2.2, Stack (LIFO)*, to show the correct number of priorities on this chip *Section 1.7.1, Initialization flow*: changed the pseudocode to a bulleted list of steps *Section 1.7.10, Examining LIFO contents*: added note "Reading the INTC_IACKRn acknowledges the interrupt request..." *Chapter 25: Enhanced Direct Memory Access (eDMA)* *Table 327: eDMA peak transfer rates (Mbyte/s)*: updated header row *Section 25.4.5, eDMA performance*: replaced cycle variable "?" with "*x*" and added an explanatory footnote *Table 331: TCD status sequence – software activated channel*: added title *Chapter 1: Direct Memory Access Multiplexer (DMAMUX)* Updated Section 20.1.2, Features *21: Clocking* *Section 21.1, Introduction*: – Removed paragraph "The system clock (CLKOUT) for..." – Updated CMU inputs to *XOSC*, *IRCOSC* and *PLL0_PHI1* (was "outputs of clock dividers") – Added "Progressive Clock Switching (PCS)" to list of items discussed in the chapter Table 209 ([SCG18.301][Covers: Saf1425]MC_CGM relationship to clocks[end]): changed Output clock of row CGM_SC_DC0 to *Core (CORE_CLK), XBAR (XBAR_CLK);* deleted row CGM_SC_DC2; changed row CGM_AC2_DC1 to *CGM_AC2_DC0* and updated description accordingly *Section 21.6.4, Sigma-Delta ADC clocking*: Removed references to multiple instances of ADC

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	Removed section "*System clock monitors*" *Figure 171 (Clock generation)*: renamed "PCFS" module to "PCS" Added Section 21.9, Progressive clock switching Chapter 27, Dual PLL Digital Interface (PLLDIG) All instances of PLL1 and PLL0 replaced with PLL Removed section "Acronyms and Abbreviations" Section 22.4, Modes of operation: removed sentence "Bypass mode is defined as the mode..."; removed text "When PLL0 is powered down..."; removed all of text starting with, "Coming out of reset..." Updated Section 22.7, Initialization information Figure 174 (PLL Control Register (PLLCR)): added footnote "This field can be written at any time..." Table 222 (PLLCR field descriptions): Modified CLKCFG field description Figure 175 (PLL Status Register (PLLSR)): updated the reset values of bits 25 and 27 to undefined (—) Section 22.6.2, Clock configuration: – updated all equations – added 'x 2' in the denominator of equation 2 *Chapter 7: Clock Monitor Unit (CMU)* Removed all references to CLKMN1 frequency less than reference clock event status *Section 7.2, Block diagram*: added text, "The full block diagram of the CMU is shown in *Figure 115*. See the "Clocking" chapter for device specific use of each CMU module." Added text, "The block diagram of the CMU module(s) is shown in the "Clocking" chapter of this Reference Manual." tagged with Block_Diagram_No *Figure 115 (Clock Monitor Unit diagram)*: The note below the diagram is moved inside the red dotted area *Table 149: CMU memory map*: renamed (was "CMU register summary") *Section 7.5, Memory map and register definition*: renamed (was "Register description and memory map") *Section 7.5.1.2, CMU Frequency Display Register (CMU_FDR)*: added introduction "The FDR is used to determine the measured..." *Section 7.5.1.3, CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)* and *Section 7.5.1.4, CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)*: added sentence "The HFREFR is configured for the high frequency reference that the CMU will use for comparing..." *Section 7.5.1.5, CMU Interrupt Status Register (CMU_ISR)*: removed FLCI from CMY_ISR register and corresponding descriptions table; added note "Before entering low-power stop..." *24: Clock Generation Module (MC_CGM)* Removed SYSCLK0 and associated connections and references *Table 225: MC_CGM register description*: merged content of "Abbreviation" and "Name" columns into single "Description" column Removed "MC_CGM memory map" table

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	*Section 24.3.1.26, Auxiliary Clock 7 Select Status Register (CGM_AC7_SS)*: update address value from 0x08E0 to 0x08E4 Deleted "MC_CGM memory map" table *Table 225: MC_CGM register description*: updated address column *Section 24.3.1.8, PCS Divider Change Register 3 (CGM_PCS_DIVC3)* to *Section 24.3.1.10, PCS Divider End Register 3 (CGM_PCS_DIVE3)*: changed "reserved" to "PLL0 PHI1" *Figure 204 (PCS Divider Change Register 3 (CGM_PCS_DIVC3))*: corrected address *Figure 205 (PCS Divider Start Register 3 (CGM_PCS_DIVS3))*: corrected address *Figure 206 (PCS Divider End Register 3 (CGM_PCS_DIVE3))*: corrected address *Chapter 27: Fast OSC Digital Interface (FXOSC)* *Table 322: XOSC configurations*: updated CTL[OSCBYP] to XOSC_CTL[OSCBYP] *Section 27.2.4, Oscillator bypass mode*: added note "The external oscillator must be running before..." *Table 324: FXOSC_CTL register field descriptions*: – Removed footnote "Software can read as well as set the OSCBYP bit." – Removed redundant table footnote "Software can read as well as clear the I_OSC bit." from OSCBYP field of CTL register Added note in *Section 27.3.1, Register descriptions* *Section 27.3.1.1, FXOSC control register (FXOSC_CTL)*: – updated reset value for M_OSC to '1' – updated reset value for M_OSC to '–' and added note referring to the "Clocking" chapter Chapter 26, IRCOSC digital interface Added new topic heading, *Section 31.1, Introduction* Updated *Section 31.2, Functional description*: *27: RAM controller (PRAM)*: *Section 27.2, Memory map and register definition*: renamed (was "SRAM controller memory map and register definition") Updated *Table 234: RAM controller memory map* *Table 235: Platform RAM Configuration Register 1 field descriptions*: renamed (was "Platform RAM Configuration Register 1 description") Chapter 28, Flash memory controller (PFLASH Controller) Section 28.4, Memory map and register definitions: renamed (was Flash memory controller memory map) *Table 4: PFCR1 field descriptions*: – Changed descriptions for fields P0_BFEN and P1_BFEN – Removed wait-state data from RWSC field description – Removed reference to "Port 1" from RWSC field descriptions Figure 7 (Platform Flash Configuration Register 3 (PFCR3)): – Added register offset and access information – Bit 17 read value changed to '0' (was undefined) – Removed references to Port 1 from BAF_DIS field description.

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	Table 9 (PFCRDE field descriptions): replaced all instances of generic Calibration Descriptor number *n* with actual number for that Descriptor *Section 1.5.1.7.1, PFlash Calibration Region Descriptor n, Word0 (PFCRDn.Word0)*: least significant 3 bits of LSTARTADDR are reserved. This is related to the PFCRDn.Word2[CRDSize] field no longer supporting sizes 1byte, 2bytes or 4bytes. – *Section 1.5.1.7.2, PFlash Calibration Region Descriptor n, Word1 (PFCRDn.Word1)*: least significant 3 bits of PSTARTADDR are reserved and are always 0. *Section 1.5.1.7.3, PFlash Calibration Region Descriptor n, Word2 (PFCRDn.Word2)*: the FCRDn.Word2[CRDSize] field no longer supports sizes 1byte, 2bytes or 4bytes. Values 00000, 00001, and 00010 are reserved. Chapter 29, Embedded Flash Memory *Section 1.2, Flash memory map and description*: added footnote "If the BAF is 8KB in size, then it is integrated with the 8KB UTEST area..." *Table 7: TestFlash block memory map*: – Removed "Manufacturer Reserved Test Flash area" row – Added rows "Flash Firmware — Minor identifier (f)", "Flash Firmware — Major identifier (F)", "Flash parameters identifiers (P)", "Flash Firmware and parameters — unused" Updated *Table 10: Flash User Registers memory map*: *Table 11: MCR field descriptions*: in PGM and ERS fields changed "UT0.AIE" to "UT0.UTE" Added *Chapter 37: Flash Memory Programming and Configuration* Chapter 32, Error Injection Module (EIM) Section 32.2.1, Memory map: renamed (was "Register summary") Table 327 (EIM memory map): specified 6-channel support Table 328 (Error Injection Module Configuration Register (EIMCR) field description): in row GEIEN changed channel range to "0–5" (was 0–3) Figure 295 (Error Injection Channel Enable (EICHEN) register): displayed 6-channel support Section 32.3, Functional description: added content. Removed sections "Modes of operation", "External signal description", "Detailed signal descriptions", "Initialization information", and "Application information" Chapter 33, Error Reporting Module (ERM) Table 333 (ERM memory map): renamed (was "Error Reporting Module"); updated columns Register and Location; renamed all sections, figures, and tables Table 335 (ERM_SR field descriptions): updated field of bit 1 to NCE0 (was "NCE00") Table 336 (ERM_EARn field description): renamed EAR field (was "EARn") Table 337 (ERM_SYNn field descritption): renamed SYNDROME field (was "SYNDROMEn")

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	*34: Analog-to-digital converter (ADC) configuration* *Section 34.1, ADC overview*: changed paragraph "All SAR and Sigma-Delta ADCs share the same supply pins (VDD_HV_ADV/VSS_HV_ADV) and ground, but have separate reference pins" to "All SAR and Sigma-Delta ADCs share the same supply pins (VDD_HV_ADV/VSS_HV_ADV), ground, as well as reference pins." *Table 280: Sigma-Delta ADC external signal description*: changed VDD_HV_ADR signal voltage value to "3–5 V" (was 5 V) *Table 286: SARB analog test channel assignment*: removed columns "Fast SAR" and "Fast SAR Channel" *Chapter 39: Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface* *Section 39.3, Features*: removed items "Left/right data alignment" and "Signed/unsigned format of converted data output" Table 357 (MCR field descriptions): updated description of field "GECEN" *Section 39.6.2.5, Request Select and Enable Register (RSER)*: updated description of field "CDVEE" *Section 39.6.2.7, FIFO Control Register (FCR)*: – Added the FOWEN control bit to enable the overwrite option for FIFO – Added FRST reset bit to provide option for FIFO flush *Section 39.7.11, Gain calibration support*: – Updated steps 7, 10, 11, 12, and note "The gain error calibration should only be done" – Added two notes: "MCR[GECEN] = 1 ensures the accurate gain error..." and "The higher the number of full-scale conversion..." *Section 39.7.12, Offset calibration support*: updated step 2 *Section 39.8, Initialization information*: changed item "Enable high-pass filter is required." to "Enable high-pass filter if required." Added *Section 39.8.1, Data conversion step* Chapter 36, Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface: *Section 1.2, Overview*: Added sentence "The number of input channels and their mapping..." to note "See the device-specific Analog-to-Digital Converters (ADC)..." Moved note "SARB implements a bias generator..." to *Figure 1 (SARADC block diagram)* *Figure 1 (SARADC block diagram)*: changed "Vddm" to "VDD_HV_ADV", "Vssm" to "VSS_HV_ADV", "VAREF1" to "VDD_HV_ADR", and "VAGND1" to "VSS_HV_ADR" Updated Section 36.4.1.1, Start of normal conversion *Section 1.4.2, Injected channel conversion*: updated "By injection trigger" item of bulleted list; added details about Enqueuing feature. *Section 1.4.10, Interrupts*: added sentence "It is recommended that conversion should not be ... bits (ECNCMR or ECJCMR) are set." *Section 1.5.1.3, Interrupt Status Register (ISR)*: Added CTUTRGERR field Added note to *Section 1.4.6, Test channel connection with internal analog channel*

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	*Section 1.4.7, External channel mapping to internal analog channel*: added sentence "After the conversion is performed on a test channel the converted data is stored at two CDR locations..." *Figure 11 (Main Configuration Register (MCR))*: – Removed EDCSELF (bit 30) – Removed footnote "Only available with normal trigger feature." – Removed footnote "Only available with injection trigger feature." *Table 6: MCR field descriptions*: removed EDCSELF (bit 30) *Chapter 1: Decimation Filter* Relocated (was chapter 40) Removed all content and references relating to CASCADE mode, EQADC, PSI mode, and BIQUAD *Table 7: DECFILTER_MCR field descriptions*: changed FTYPE[11] description to "Reserved" Removed section "Triggered Output" Chapter 38, Temperature Sensor: unchanged Chapter 39, System Timer Module (STM): unchanged *Chapter 31: Software Watchdog Timer (SWT)*: unchanged *Chapter 45: Periodic Interrupt Timer (PIT)* Relocated to *Timers* section *Figure 495 (Block diagram of the PIT)*: removed RTI block and associated connectivity *Table 597: PITx memory map*: removed "Reset value" column; added reserved rows 0000_0004–0000_00DF and 0000_00E8–0000_00FF Chapter 42, CAN Subsystem *Section 1.3.3, Dual clock sources*: modified note "Stop this module when the device is in STOP mode..." *Section 1.3.5.2.11, Timeout Counter Value Register (TOCV)*: corrected the register access from "R/W" to "W1C" *Section 1.3.5.2.25, Rx FIFO 0 Configuration Register (RXF0C)*: corrected the F0S bit field length from 10–15 to 9–15 *Section 1.3.5.2.26, Rx FIFO 0 Status Register (RXF0S)*: added new bits F0PI 10–15 *Section 1.3.5.2.30, Rx FIFO 1 Status Register (RXF1S)*: added new bits F1PI 10–15 *Section 1.3.5.2.32, Rx Buffer / FIFO Element Size Configuration (RXESC)*: corrected the register access from "R" to "RP" *Section 1.3.5.2.34, Tx FIFO/Queue Status Register (TXFQS)*: added new bits TFGI 19–23 *Section 1.3.5.2.44, Tx Event FIFO Status Register (TXEFS)*: added new bits EFGI19–23 Updated *Section 1.3.13.3, Tx FIFO* Updated *Section 1.3.13.5, Mixed dedicated Tx Buffers / Tx FIFO* Updated *Section 1.3.5.2.6, CC Control Register (CCCR)* Removed figure "Message RAM Configuration" Removed table "Standard ID Filter Configuration register field descriptions"

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	Chapter 43, Serial Interprocessor Interface (SIPI) *Section 48.1.1, Scalability*: Added bulleted list for the last paragraph that starts, "The rest of this section..." *Section 48.2, Overview*: removed the sentence, "The AHB Master interface..." *Figure 669, SIPI module*: changed sentence "Channel 2 is given the highest priority for transmission, and is the only channel that is capable of..." to "Channel 0 is given the highest priority for transmission, and channel 2 is the only channel that is capable of..." *Section 48.6.2, Frame format*: added new content. Retained the original paragraph. *Section 48.7, Transfer types*: added the sentence, "The Debug SIPI frame is inserted..." *Figure 680 (Read answer transfer (LFAST frame encapsulation is not shown))*: changed title (was "Read answer transfer") *Figure 682 (Register write transfer (showing LFAST frame encapsulation))*: changed title (was "Register write transfer") *Figure 683 (Write transfer (LFAST frame encapsulation is not shown))*: changed title (was "Write transfer") *Figure 685 (Write Acknowledge transfer (LFAST encapsulation is not shown))*: changed title (was "Write Acknowledge transfer") *Figure 686 (ID request transfer (LFAST encapsulation is not shown))*: changed title (was "ID request transfer") *Section 48.8, Transfer API and flow charts*: removed note "Read answer ipi_int_readx interrupt signals are also cleared..." Section 43.11, Errors: – Bullet 2: changed 'CRC IP' to 'CRC engine' – Removed note "Refer to the Cyclic Redundancy..." *Table 854, SIPI memory map*: updated the column 'Offset from SIPI base' to show only 16 bits instead of 32 bits *Section 48.16.1.1, SIPI Channel Control Register (SIPI_CCRn)*: updated "channel 2 – channel 0 – channel 1 – channel 3" to "channel 0 – channel 1 – channel 2 – channel 3"; modified Note text *Table 857, SIPI_CCRn field descriptions*: changed description of field "CHEN" *Section 48.16.1.8, SIPI Module Configuration Register (SIPI_MCR)*: – Updated footnote for the MCR[AID] in the register table – Updated register names "SIPIMCR" to "SIPI_MCR", "SIPIACR" to "SIPI_ACR", "SIPIERR" to "SIPI_ERR", "SIPIARR" to "SIPI_ARR", "SIPIMAXCR" to "SIPI_MAXCR", and "SIPISR" to "SIPI_SR" *Chapter 48: Deserial Serial Peripheral Interface (DSPI)* *Figure 683 (DSPI block diagram)*: removed "MTRIG" signal *Section 48.1.2, Features*: removed "support for parallel and serial chaining of several DSPI modules inside MCU" *Table 749: DSPI signal description*: removed "MTRIG" signal and "Slave mode: Master Trigger" description *Section 48.2.3, PCS4 — Peripheral Chip Select 4*: removed sentence "In slave mode, the active low MTRIG is an output trigger signal that indicates that a change in data to be serialized..." *Table 750: DSPI memory map*: added footnote "This register is not used on the SPC572Lx device, refer to the device configuration chapter for details on..." for registers DSPIx_PISR0 to DSPIx_PISR7

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	*Figure 696 (DSPI DSI Configuration Register 0 (DSPI_DSICR0))*: bit 0 set to reserved (was MTOE) and bits 2-7 set to reserved (was MTOCNT[5:0]) *Section 48.5.3, Deserial Serial Interface (DSI) configuration*: removed sentence "The DSPI also supports parallel chaining allowing several DSPIs and off-chip SPI devices..." *Section 48.1.4.2, Slave mode*: removed sentence "If DSICR0[CID] is set and the data in the DSPI_COMPR1/0 differs from..." *Section 48.5.3.5.2, Change in data control*: removed sentence "The MTRIG output signal is asserted every time a change in data is detected." Removed section "Multiple Transfer Operation (MTO)" Chapter 45, LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications *Table 949: DFSR field descriptions*: – Added reserved rows for Reserved bits 16–17 – Swapped LDSLPS and LRSLPS descriptions Section 45.6.2.1, Mode Configuration Register (MCR): added figure footnote, "Writable only once after the asynchronous reset." This footnote is now used as the reference for the MCR[MSEN] field. *Table 953: PLLLSR field descriptions*: swapped the field description and bit field value descriptions for LDPDS and LRPDS. *Section 50.7.2.2.1.1, Hunt correlation mode (COCR[CORRMD] = 00b)*: removed "(COCR[CORRMD] = 00b" from title *Section 50.6.2.15, LVDS Control Register (LCR)*: updated reset value to 0000_5024h (was 0000_5004h). Chapter 46, Fast Ethernet Controller (FEC) Added footnote "Interface selection is chip-specific..." to *Section 1.1.2, Features*, *Section 1.2.2, Interface options*, *Figure 1 (FEC block diagram)*, *Figure 18 (Receive Control Register (RCR))*, *Table 5: FEC signal descriptions (for FEC with RMII)* and *Table 25: RCR field descriptions* *Section 1.4, Memory map and register definition*: deleted "You control the FEC by writing into..."; added "All accesses to and from the registers must be via 32-bit accesses..."; added "Accessing reserved or unimplemented register spaces in the FEC will not return..." *Section 1.4.63, Receive Descriptor Ring Start Register (ERDSR)*: added "You should write zeros to bits 1 and 0. Hardware ignores..."; changed "This register is not reset" to "This register is undefined at reset"; changed all RMON and IEEE registers from read-write to read-only. *Section 1.5.8, Network interface options*: RCR[RMII_MODE] bit setting changed to "=1" (was "cleared") for RMII enable setting description *Chapter 1: SENT Receiver (SRX)* *Section 1.3.2.4, Fast Message Ready Status Register (FMSG_RDY)*: changed sentence "register will get asserted on Slow Message reception" to "register will be asserted on Fast Message reception." *Section 1.3.2.5, Slow Serial Message Ready Status Register (SMSG_RDY)*: changed sentence "register will get asserted on Fast Message reception" to "register will be asserted on Slow Message reception."

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	Chapter 48, LINFlexD Updated *Section 1.3.2.5.2, Bit error* *Section 1.3.2.5.6, Timeout error*: reorganised and updated contents *Section 1.3.2.5.7, Noise*: content and figures reorganized Updated *Section 1.3.4.5, 13-bit frames* Updated values and descriptions for fields SADDR[31:0] and DADDR[31:0] in tables: *Table 10: TCD setting — Master node — Tx mode*, *Table 11: TCD setting — Master node — Rx mode*, *Table 13: TCD setting — Slave node — Tx mode*, *Table 15: TCD setting — Slave node — Rx mode*, *Table 16: TCD setting — UART — Tx mode* *Table 16: TCD setting — UART — Tx mode*: updated value for field SLAST[31:0] *Table 19: LINCR1 field descriptions*: set field "SLFM" to reserved *Section 1.4.2.4, LIN Error Status Register (LINESR)*: updated description of field "OCF" *Section 1.4.2.9, LIN Time-Out Control Register (LINTOCR)*: updated description of field "IOT" *Section 1.3.4.8, UART receiver*: added note "When WLS bit = 1, WL = 0 or 1 should not be used..." *Table 18: LINFlexD memory map*: added footnote "Refer to the device configuration chapter for..." *Section 1.4.2.3, LIN Status Register (LINSR)*: changed "AUTOSYNC_COMP" bit from "read only" to "w1c" *Section 1.4.2.7, LIN Time-Out Control Status Register (LINTCSR)*: added note "For proper functionality..." to description of field CNT *Table 48: Interface signals*: Replaced "out_ps_clr_DRF" with "out_ps_DRF", "out_ps_clr_DTF" with "out_ps_DTF", "out_ps_clr_PE[3:0]" with "out_ps_PE[3:0]", "out_ps_clr_FEF" with "out_ps_FEF", "out_ps_clr_BOF" with "out_ps_BOF", "out_ps_clr_TOF" with "out_ps_TOF"; added footnotes for all PSI5-S UART interface signals. Added *Section 1.3.2.7.1, Start detection mechanism* Added *Section 1.3.2.7.2, Delimiter detection mechanism* *Figure 35 (UART — TX memory map)*: inserted "1 DMA TX channel/filter (TCD single and/or linked chain)" label at bottom of figure *Figure 37 (UART — RX memory map)*: inserted "1 DMA RX channel/filter (TCD single and/or linked chain)" label at bottom of figure *Section 1.4.2.9, LIN Time-Out Control Register (LINTOCR)*" updated description of field HTO *Section 1.4.2.13, LIN Control Register 2 (LINCR2)*: updated access to fields WURQ, DDRQ, DTRQ, ABRQ, and HTRQ from "read only" to "read/write" *Section 1.4.2.22, Global Control Register (GCR)*: added table in description of field SR *Chapter 6: Reset Generation Module (MC_RGM)* *Figure 1 (MC_RGM block diagram)*: added "destructive reset from SSCM" to "'Destructive' Reset Filter" block Removed Table (MC_RGM memory map) *Figure 2 ('Destructive' Event Status (RGM_DES) register)*: added "F_SSCM_DEST" to bit 17 MSB (bit 14 LSB)

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	Table 818 ('Destructive' Event Status (RGM_DES) register Field Descriptions): added field "F_SSCM_DEST" *Figure 3 ('Destructive' Event Reset Disable (RGM_DERD) register)*: removed all registers except D_PORST and D_TSR_DEST Table 819 ('Destructive' Event Reset Disable (RGM_DERD) register Field Descriptions): added field "D_SSCM_DEST" *Figure 5 (Destructive' Bidirectional Reset Enable (RGM_DBRE) register)*: – Updated register address information to "0x030" was (0x010) – Added field "D_SSCM_DEST" – Added field description for "BE_JTAG_DEST" *Figure 7 ('Functional' Event Reset Disable (RGM_FERD) register)*: bit 31 set to reserved (was ESR_0) *Table 15: 'Functional' Event Reset Disable (RGM_FERD) register field descriptions*: removed field "ESR_0" *Figure 8 ('Functional' Event Alternate Request Register (RGM_FEAR))*: D_ESR0 bit set to reserved *Table 16: 'Functional' Event Alternate Request Register (RGM_FEAR) field descriptions*: removed field "D_ESR0" *Table 17: 'Functional' Bidirectional Reset Enable (RGM_FBRE) register field descriptions*: added description for field "BE_PLL0" *Figure 9 ('Functional' Bidirectional Reset Enable (RGM_FBRE) register)*: changed reset value of field "BE_PLL0" from 17 to 1 Chapter 50, Boot Assist Flash (BAF) Table 837 (BAF memory organization): – For "BAF data area and stack", new value "4000_0000h – 4000_01FFh" added – Changed Data and stack area 5280_0000h to 5280_01FFh Section 50.3.2, Device initialization: bullet 4: added "SRAM area from 4000_0000h to 4000_03FFh" and changed BAF scratch memory requirement to 512 bytes Updated Figure 873 (Flow of control) Section 50.3.4.1, 'Soft' DCF clients: added Serial Boot DCF client; added paragraph "The serial boot DCF client provides... Table 841 (BAF DCF client list): added Serial Boot DCF client address Section 50.3.4.2.3, Security Watchdog Service Address (SEC_IAC8): removed instance of "IOP" from Warning text Added Section 50.3.4.6, BAF Serial Boot DCF Client -- SER_BOOT_CBACK Added Table 844 (Serial boot callback function return value definition) Updated Section 50.3.6, Optionally perform a serial boot Updated Section 50.3.7, UART serial boot configuration Section 50.4, Resources: removed "PFLASH" bulleted list item Chapter 51, System Status and Configuration Module (SSCM) Section 51.3.1.1, System Status Register (STATUS): changed bits 5 and 6 Section 51.4.1, DCF mechanism: removed paragraphs "The SSCM maintains three DCF record areas..." and "Which of those areas are scanned..."

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	*Chapter 1: Power Management Controller digital interface (PMC_dig)* All registers in chapter updated in line with new Voltage Detector setup on device *55: Power Control Unit (MC_PCU)* *Table 806: MC_PCU register description*: merged columns "Name" and "Description" into column "Register" Removed table "MC_PCU memory map" *Chapter 1: Mode Entry Module (MC_ME)* *Table 3: MC_ME register description*: merged columns "Name" and "Description" into column "Register" Removed table "MC_PCU memory map" *57: e200z215An3 Core Debug Support* *Section 57.5.3, JTAG/OnCE pins*: changed sentence "Asserting dbg_dbgrq will cause the chip to..." *Section 57.5.7, Access to debug resources*: added text "via the OCR_{WKUP} control bit" to paragraph "Some cases of write accesses..." *Chapter 65: Buddy Device—Debug and Calibration Interface (DCI)* The mux sequences of various JTAG sources updated *Section 65.3.1, DCI control register (DCI_CR)*: – Renamed field "FPD_FUNC" (was "FPD_ESR0") – Renamed field "FPD_DEST" (was "FPD_PORST") *Chapter 66: Buddy Device—JTAG Controller (JTAGC)*: unchanged Chapter 58, IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG) *Section 1.7, APU (Advanced Protocol Unit) Operation*: renamed (was "APU (Advanced Protocol Unit") Added *Section 1.8, Functional Description* *Section 1.6.2.3.15, Auxiliary Pin Function Control (APFC[1:0])*: corrected APFC encoding value *Chapter 68: JTAG Data Communication (JDC)* Added reset source information for each register *Table 1561: JTAG instructions*: renamed (was "6-bit JTAG instructions"); updated content *Chapter 59: Buddy Device—JTAG Master (JTAGM)* *Table 1346: JTAGM_MCR register field descriptions*: added note "After being written to..." in descriptions of fields "SWRESET", "evti1_assert" and "TCKSEL" *Section 59.5.1.2, Status register (JTAGM_SR)*: added note in description of field NR Chapter 61, Sequence Processing Unit (SPU) Removed all content and references relating to NXMC and EVTI/O 1 Section 61.5, Register definition: removed C0PEVP register; added C2PEVP register; removed C0PIS register; added C2PIS register

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	*Figure 1314 (SPU register summary)*: corrected Counter Compare Status (CCOMS) and Counter Capture Status (CCAPS) register names *Section 61.10.1.6, SLU status (SS) register*: bit 20 set to reserved (was EM0); bit 22 set to *EM2* (was reserved) *Table 1332: SPU action decoder*: – From client group counter bit: counter ID 00000 set to reserved (was CPU0); counter ID 00010 set to CPU2 (was reserved); counter IDs 10000 and 10001 set to reserved (was NXMC1,2) – From client group action bit: removed PMC1,2,3,4 and reference to PMC in comments for action IDs 0101 to 1000 – From client group enable bit: removed reference to PMC – From system group action bit: removed Timestamp0,1_1,2 and references to AHB in comments for action IDs 0011 to 0110 *Section 61.10.1.9.5, Trace group configuration registers (DTMGC, PTMGC, OTMGC and WTMGC)*: bit 0 set to reserved (was CPU0); bit 2 set to *CPU2* (was reserved) *Section 61.10.1.10, Counters control n (CCTRLn) registers spu_cntr[0-15]_cntrl_reg (SPU Counter[0-15] Control Register)*: removed Timer reference from register name to maintain the naming consistency *Section 61.10.1.11, Counters compare n (CCMPn) registers spu_cntr[0-15]_cmpr_reg (SPU Counter[0-15] Compare Registers)*: removed Timer reference from register name *Section 61.10.1.12.1, Counter compare status (CCOMS)*: added note explaining when w1c should be performed Added Section 61.6.8, Format for CKSRC & CKDATA Values *Table 1339: CCTRLn register field descriptions*: removed sentence "The mapping of the PMC inputs with the counters is shown in Figure 1426." from description of field COUNT_INCR *Table 1340: CCMPn register field descriptions spu_cntr[0-23]_cmpr_reg*: changed table title (was CCTRLn...) *Chapter 68: Development Trigger Semaphore (DTS)*: unchanged Chapter 63, Nexus Aurora Router (NAR): unchanged *66: e200z215An3 Nexus 3 Module*: no substantial changes Chapter 65, GTM Development Interface (GTMDI) Section 65.4.1.1, Device identity register (DID): updated contents of bits 11–1 *Chapter 1: Register Protection (REG_PROT)* *Section 1.4.2.4, Global Configuration Register (GCR)*: updated introduction Chapter 67, Password and Device Security Module (PASS) Section 67.3, Memory map and register definition: changed "*Chip Select (CS[14:0])*" to *"Client Select (CS[14:0])"* Table 1157 (PASS memory map): – Changed title (was Module memory map) – Changed Challenge Status Register abbreviation to CHSTAT (was CSTAT) and added its reset value – Removed "Censorship", "Production Disable" and "Clock Jitter Enable" rows

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	Section 67.3, Memory map and register definition: added "Do not write to address 0x00D0 with bit value 0." Section 67.3.1.1, Life Cycle Status register (LCSTAT): Rewrite of second paragraph, describing the way that the register reset value is obtained Figure 1221 (Life Cycle Status register (LCSTAT)): CNS field reset value changed to *x*; LIFE field reduced to a 3-bit field and reset value altered. Table 1158 (LCSTAT register field descriptions): changed LIFE field description to reflect 3-bit dimension Section 67.3.1.5, Password Group n - Lock 0 Status register (LOCK0_PGn): removed last two paragraphs relating to the *BAF PLock* bit Figure 1225 (LOCK0_PGn register): replaced ATSL with reserved field and added corresponding footnote Table 1162 (LOCK0_PGn register field descriptions): replaced ATSL with reserved field Figure 1228 (LOCK3_PGn Register): Added footnote - *Not all devices...* Table 1165 (LOCK3_PGn register field descriptions): Changed descriptions for fields RL0 to RL4 and added *reserved* field descriptions Removed section "Hardware Interlock (HI)" Removed section "Clock Jitter Enable" Moved all DCF client information into a separate section, Section 67.4, DCF clients Removed section *Life cycle and Security* Section 67.4.1, Censorship: repositioned; added titles to register description figure and field descriptions table. Section 67.4.2, Production Disable: repositioned; reworded, removed "Hardware Interlock" from content. Section 67.4.3, Debug Disable (DD): repositioned; removed all content relating to Clock Jitter Figure 12 (Production Disable DCF client): removed "HI" bit (is now reserved) Table 17 (Production Disable DCF Client field descriptions): removed "HI" bit field description (is now reserved) Added Section 67.5.2, Debug Interface Access Added Section 67.5.3, Flash Memory Read Protection Removed Section Unlocking the debug interface via JTAG Section 67.6.2, Modifying Life Cycle (maturing the device): complete section rewrite Section 67.6.3, Setting lock bits in a password group: changed *"...the DCF record has to be added to...* to *"...the DCF record has to be added as shown in..."* Table 1170 (CS/ADDR Values for Lock Registers): DCF CS/ADDR column values changed and corresponding explanatory footnote added Table 1171 (Setting some bits in the MIDLOCK field): removed *Data record: OEM production* row *Chapter 86: Wakeup Unit (WKPU)* Reinstated *Figure 1977 (NMI Configuration Register (NCR))* *Section 86.2.2, Features*: added bullet about I/O external interrupts Section 68.3, Memory map and register description: added details for registers "WISR", "IRER", "WIREER", and "WIFEER"

Table 1167. Revision history(Continued)

Date	Rev	Changes
03-Apr-2013	2 (cont'd)	Added *Section 86.5.2, External Wakeups/Interrupts* *Chapter 1: GTM101 IntegrationWrapper (GTMINTGTM-WRAP) Module*: Added "0" prefix to reset value of selected registers Updated ST disclaimer
27-Feb-2015	3	Editorial and formatting updates Chapter 1, Preface: unchanged Chapter 2, Introduction: no substantial updates *Chapter 3: Embedded memories*: unchanged *4: Signal description*: unchanged *5: Memory map*: unchanged *6: Device configuration* *Table 22: INTC implemented registers*: – Changed "INTC Software Set/Clear Interrupt Register" abbreviations to "INTC_SSCIR0–INTC_SSCIR31" (were "INTC_SSCIR0_3–INTC_SSCIR28_31") to reflect the SSCIRs redefinition to be 8 bits (instead of 32 bits) in size – Changed "INTC Priority Select Register" abbreviations to "INTC_PSR0–INTC_PSR1023" (were "INTC_PSR0_1–INTC_PSR1022_1023") *Table 23: Interrupt sources* : Major updates *Table 29: DMAMUX memory map*: generated combining contents from two separate tables; added two footnote "Offsets are from DMAMUX_0 base address..." and "Each channel must be configured separately via setting..." *Section 6.6.1, Software Watchdog Timer (SWT) configuration*: removed "Programmable selection of system or internal..." item from SWT features list *Section 6.7.1, FEC interface*: added sentence "The MII Management Frame Register (MMFR) is not supported on SPC572Lx." *Table 43: CAN bus controllers*: removed column "CAN FD support" Added *Section 6.7.9, BAF configuration* *Table 59: Protected SIUL2 registers*: added registers "MSCR 936-967" *Table 66: Protected MC_RGM registers*: Replaced "PSRTn" with "PSRT0-7" *Figure 26 (PASS_LOCK0_PGn register)*: updated "Read lock region 0" to Reserved *7: Reset and Boot* *Section 7.3.3, PHASE0 Phase: analog supply initial configuration*: – In bulleted list "To exit PHASE0 and enter PHASE1": changed "Core low voltage detect..." to "Gating PHASE0 must be released"; – After sentence "This phase is typically completed..." added "the device is guaranteed to remain in this phase for at least 0.5 µs"

Table 1167. Revision history(Continued)

Date	Rev	Changes
27-Feb-2015	3 (cont'd)	*Chapter 9: Device Configuration Format (DCF) Records* *Table 110: Series of DCF records in UTEST flash memory*: changed address offset "8n-1" to "8n" and "8n" to "8(n+1)". *Figure 44 (Appending DCF records)*: removed "overwrite" and arrow. Revised testin Warning immediately before this figure and in text immediately after it. *Section 9.5, DCF client table*: added "UTEST" in Parity description *Table 112: DCF client list*: • Removed "Flash special strategy" column • Added "STCU" section • Updated client "Censorship" row • Changed LOCKn_PGm reset values to 0xFFFF_FFFF • Changed IPS read direct access to 'no' for LOCKn_PGm clients • Removed client "Life Cycle (LCSTAT)" row • In BAF 'Soft' Clients section, changed DCF addresses from hexadecimal to binary *Section 9.5.3, Miscellaneous DCF registers*: – Set bit 16 to "*OPT_RGM_ESR0_HW" (was Reserved)* and bits 13–15 to "*XOSC_IBIAS_SEL" (were Reserved)* – Added note "To know the capacitance values relative to the 5-bit binary combinations" to XOSC LOAD CAP SEL[0:4] field description – Removed register "PMC_REE" *Chapter 10: Power management* Throughout chapter: removed all references to "HVD"; removed "[DEST]" from phase names Updated *Figure 51 (Internal LVD and HVD in configuration example during power-up)* *Table 119: POR and voltage monitors description[[Lavaredo]]*: Updated and restructured content. *Table 121: Voltage monitors configurability[[Lavaredo]]*: Updated and restructured content. Updated *Figure 53 (Example of VDD_LV power-up sequencing)* Removed figure: *Threshold variation during power-up sequence: hot point LVD disabled*. *Chapter 11: Security* *Section 11.1, Basic security*: replaced word 'BAM' with 'BAF' *11: Calibration and Debug*: unchanged *12: Core e200z215An3 description*: unchanged *Chapter 16: System Integration Unit Lite2 (SIUL2)*: unchanged *Chapter 17: Crossbar Switch (XBAR)* Throughout chapter: removed all references to the following register instances: "XBAR0_PRS3", "XBAR0_CRS3", "XBAR0_PRS4", "XBAR0_CRS4", "XBAR0_PRS5", "XBAR0_CRS5", "XBAR0_PRS6", "XBAR0_CRS6", "XBAR0_PRS7", "XBAR0_CRS7" *Chapter 20: Peripheral Bridge*: unchanged

Table 1167. Revision history(Continued)

Date	Rev	Changes
27-Feb-2015	3 (cont'd)	*Chapter 7: System Memory Protection Unit (SMPU)*: unchanged Chapter 17, Intelligent AHB Gasket (IAHBG): unchanged Chapter 18, Interrupt Controller (INTC) *Section 1.1, Introduction*: changed number of priorities from 16 to 64 *Section 1.2, Block diagram*: revised the stem sentence and figure title to account for all possible processor combinations *Section 1.5.2.3, INTC Current Priority Register for Processor n (INTC_CPRn)*: changed "PRI" field width from 4 to 6 to encompass bits 26–31 *Section 1.5.2.6, INTC Software Set/Clear Interrupt Registers (INTC_SSCIR0–INTC_SSCIR31)*: changed "... can write to the CLR bit and not the SET bit, otherwise ..." to "... can write to the CLR bit, otherwise ..." *Section 1.5.2.7, INTC Priority Select Registers (INTC_PSR0 – INTC_PSR1023)*: changed "PRIn" field width from 4 to 6 to encompass bits 26–31 *Section 1.7.5.2.2, Raised priority preserved*: • Deleted the text "Also, during the epilog of the interrupt exception handler..." • Replaced the timing diagram and associated table of event descriptions. Completely replaced *Section 1.7.10, Examining LIFO contents* *Chapter 25: Enhanced Direct Memory Access (eDMA)* *Section 25.3.24, TCD Transfer Attributes (DMA_TCDn_ATTR)*: revised the SSIZE field description *Table 326: DMA_TCDn_CSR field descriptions*: changed "XBS" to "XBAR" in "BWC" field description. *Chapter 1: Direct Memory Access Multiplexer (DMAMUX)* Throughout chapter: replaced "DMA Mux" with "DMAMUX"; changed Channel Configuration registers to "CHCFG0–CHCFG15" (were "CHCONFIG0–CHCONFIG15") Renamed chapter (was "DMACHMUX") Updated *Section 1.2.3, Modes of operation* Removed section "Overview" *Section 1.4, Memory map/register definition*: – Renamed (was "Memory map and register definition") – Updated introduction *Table 8: DMAMUX memory map*: – Renamed (was "DMACHMUX memory map") – Updated entirely Removed Section "Register descriptions" and left its subsections Removed table "Channel and trigger enabling" Updated *Section 1.6.2, Enabling and configuring sources* *Chapter 21: Clocking* *Figure 173 (Clock distribution)*: updated clock source for BIU on the GTM to XBAR_CLK from PBRIDGEx_CLK. Added *Section 21.5.1.4, PLLDIG initialization information*.

Table 1167. Revision history(Continued)

Date	Rev	Changes
27-Feb-2015	3 (cont'd)	Added *Section 21.6.5, Successive Approximation Register ADC clocking*. *Table 197: Maximum system level clock frequencies*: Changed the maximum frequency of ADCs from 16 to 14.6 MHz. *Chapter 27: Dual PLL Digital Interface (PLLDIG)* Table 224 (PLLDV field descriptions): updated binary combination definitions in field description RFDPHI *Section 27.10.2, Clock configuration*: – Updated f_{pll_phi1} equation—removed factor of 2 to denominator – Updated f_{pll_VCO} equation—removed factor of 2 to numerator – Updated f_{pll_VCO} equation—removed PLL1DV[PREDIV] from denominator – Removed f_{pll_VCO} equation which shows the equality with f_{pll_ref} frequency *Chapter 7: Clock Monitor Unit (CMU)* *Section 7.6.3, CLKMN1 supervisor*: added notes for LFREF and HFREF programming considerations Table 231 (CMU_ISR field descriptions): added note to the "OLRI" field description, "While entering STOP mode, OLRI may be triggered. To avoid this when the system is..." *24: Clock Generation Module (MC_CGM)*: unchanged *Chapter 27: Fast OSC Digital Interface (FXOSC)* *Table 322: XOSC configurations*: – Removed the row with the first three column elements 'X, 1, X'. – Added the row with the first three column elements '1, 1, X'. – Added the row with the first three column elements '0, 1, X'. – Changed "No crystal, high-impedance" to "Crystal, or left opened" in XTAL column – Removed row with last four column elements "ground, external clock, XTALOUT, and Normal OSC Enabled" *Section 27.3.1, Register descriptions*: removed note "Module generates transfer (ips_xfr_err) only..." *Section 27.2.2, Oscillator startup delay*: revised first paragraph Chapter 26, IRCOSC digital interface *Section 31.3.1, Register descriptions*: removed note "Module generates transfer error..." *Section 31.3.1.1, IRCOSC Control register (IRCOSC_CTL)*: changed bit 7 to RW from RO *27: RAM controller (PRAM)*: unchanged Chapter 28, Flash memory controller (PFLASH Controller) *Section 1.5.1.7.3, PFlash Calibration Region Descriptor n, Word2 (PFCRDn.Word2)*: set bits 0–15 to "Reserved" (were fields "M0EN–M15EN") Chapter 29, Embedded Flash Memory (MP55): substantial changes

Table 1167. Revision history(Continued)

Date	Rev	Changes
27-Feb-2015	3 (cont'd)	*Chapter 29: Decorated Storage Memory Controller (DSMC)*: unchanged *Chapter 37: Flash Memory Programming and Configuration* *Section 37.4.1, Implementing secure read protection*: changed the relevant fields from RL4–RL0 to RL2–RL0 Chapter 32, Error Injection Module (EIM): unchanged Chapter 33, Error Reporting Module (ERM): unchanged *34: Analog-to-digital converter (ADC) configuration* Removed Section "SARB interface to PMC" *Chapter 39: Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface* *Table 507: RSER field descriptions*: updated description of field "DFFDIRE" *Table 506: SFR field descriptions*: updated description of fields "DFFF", "WTHL", "WTHH", and "DFFF" *Table 508: OSDR field descriptions*: in OSD field description, added note "Refer to the tsettling and tLATENCY parameters in the device datasheet for the..." *Table 509: FCR field descriptions*: updated description of field "FTHLD" *Section 39.7.8, Data conversion*: – Added description of "FIFO Flush Option" – Added sentence "Refer to the tsettling and tLATENCY parameters in the device datasheet for the minimum allowed..." Chapter 36, Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface *Section 1.4.3, Abort conversion*: revised information on ABORT/ABORTCHAIN *Section 1.4.4.1, Conversion timings*: revised information on precharge duration Added *Section 1.4.14, Stop Mode* Added information on "WSEL_CH" field parameter dependency in *Section 1.5.1.18, Internal Channel Watchdog Select Registers (ICWSELR0–ICWSELR11)*, *Section 1.5.1.28, Test Channel Watchdog Select Registers 0–3 (TCWSELR0–TCWSELR3)*, and *Section 1.5.1.39, External Channel Watchdog Select Registers (ECWSELR0–15)* Corrected description of field "CDATA" in Table 394 (ICDR0–95 field descriptions), Table 422 (TCDR96–127 field descriptions), and Table 442 (ECDR128–255 field descriptions) field descriptions) Added *Section 1.7, Initialization Information* *Chapter 1: Decimation Filter*: unchanged Chapter 38, Temperature Sensor *Section 4.3.1, Temperature threshold detection (digital output generation)*: changed time to update digital outputs from 1 µs to 1 ms Updated *Figure 8 (Temperature formula)*

Table 1167. Revision history(Continued)

Date	Rev	Changes
27-Feb-2015	3 (cont'd)	Chapter 44, CAN Subsystem: substantial changes Chapter 45, Serial Interprocessor Interface (SIPI) *Section 48.10.1, Initialization mode (INIT)*: removed paragraph "When in initialization mode, all message transfers are stopped. If software invokes..." *Section 48.10.3, Module disable (MD)*: added information to disable Rx functionality *Table 865: SIPI_MCR field descriptions*: revised "MOEN" field description to say that future SIPI Tx transfers are disabled when this bit is negated *Table 870: SIPI_ERR field descriptions*: revised binary combination descriptions for field TOE3–TOE0 *Chapter 48: Deserial Serial Peripheral Interface (DSPI)* Throughout chapter: removed all instances of registers "PISR" (PISR0–PISR7) and relative content; removed "DMA request" feature content; removed DOZE mode content. *Section 48.1.2, Features*: removed features "DMA request support for DDIF" and "edge sensitive hardware trigger"; removed text "support for doze mode". *Section 48.1.4.4, External Stop mode*: added content relating to External Stop mode requests in DSPI Slave mode. *Table 749: DSPI signal description*: added footnote "This pin is not available..." *Table 750: DSPI memory map*: added reset value of register "DSPIx_SR": "0201_0000h" *Section 48.3.1, DSPI Module Configuration Register (DSPI_MCR)*: – Set bit 16 to "Reserved" (was field "DOZE") – Removed text "parameterized, with a default reset value of" from "MDIS" field description *Table 751: DSPI_MCR field descriptions*: in "MTFE" field description, added sentence "MTFE should not be set in ITSB..." *Section 48.3.4, DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTARn_SLAVE)*: renamed field relative to bit 9 to "FMSZ5" (was "FMSZ[5]") *Figure 690 (DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER))*: removed field "DDIF_DIRS" from bit 17 *Figure 696 (DSPI DSI Configuration Register 0 (DSPI_DSICR0))*: – Added footnote to "TPOL" field "This bit is reserved if the device does..." – Removed field "TRRE" from bit 13 – Renamed fields relative to bits 1 and 8 to "FMSZ4" and "FMSZ5" respectively (were "FMSZ[4]" and "FMSZ[5]") Updated *Table 765: DSPI_DSICR0 field descriptions*

Table 1167. Revision history(Continued)

Date	Rev	Changes
27-Feb-2015	3 (cont'd)	*Section 48.3.17, DSPI DSI Configuration Register 1 (DSPI_DSICR1)*: – Renamed field "TSBCNT[5:0]" to "TSBCNT" – Updated field "TSBCNT" width from 6 to 5 to encompass bits 3–7 – Set bit 2 to Reserved – Updated the access value of field "DSI64E" from RW to RO *Section 48.5.3.4, DSI deserialization*: removed expressions "or DMA request" and "RSER[DDIF_DIRS]" *Figure 723 (DSI serialization diagram)*: removed Parallel Input Select Registers 0-3 block from diagram *Section 48.5.9.1, MSC Dual Receiver Support with PCS Switchover*: added note "When using Dual receiver support, it must be ensured..." *Table 799: Interrupt and DMA request conditions*: In row "DSI Deserialized Data Match" changed Request Type DMA to "—" (was Yes) *Section 48.5.13, Power saving features*: added note "When in Slave Mode..." *Section 48.5.13.1, Stop mode (External Stop Mode)*: added paragraph "If a DSPI block is configured in Slave mode..." Chapter 47, LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications Removed reference to 312/320 MHz from entire chapter and referred to as high data rate. *Section 50.6.2.15, LVDS Control Register (LCR)*: – Field "LVRXOP" (bits 26–28) is now split into 3 individual bits: LVRXOP_TR (bit 26), Reserved (bit 27), and "LVRXOP_BR" (bit 28) – Swapped the binary combination descriptions of field "LVRXOP_TR" – Updated field description "LVCKSS" *Section 50.7.7.1.2, Data receive*: in the first paragraph, changed "MCR[TXEN]" to "MCR[RXEN]". Chapter 48, Fast Ethernet Controller (FEC): unchanged *Chapter 1: SENT Receiver (SRX)* Throughout chapter: – In register description sections of selected registers, changed note "The following register figure..." to "Reads of bits beyond those for the supported number..." – Removed references to "SRX_DATA_CTRL2" register In Section 1.1.1, Features, changed bulleted list item "Supports compensation ... for drift and jitter in the Transmr clocks per channel" to "Supports compensation... for drift in the Transmitter and Receiver clocks per channel" In Section 1.1.2.2, Low power modes, added two notes: "If the SOC requests entry to debug or stop mode..." and "The message read registers..." In *Section 1.3.2.1, Global Control Register (GBL_CTRL)* – Added field "NIB_LEN_VAR_LIMIT" at bit 15 – Updated binary combination descriptions of field "TSPRSC" *Section 1.3.2.4, Fast Message Ready Status Register (FMSG_RDY)*: added note "These bits are either "write 1 to clear" (w1c) or self-clearing..." in "F_RDYn" field description.

Table 1167. Revision history(Continued)

Date	Rev	Changes
27-Feb-2015	3 (cont'd)	*Section 1.3.2.6, Data Control Register 1 (DATA_CTRL1)*: added note "Reset value changes depending on the number of channels that are enabled on..." *Section 1.3.2.15, DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)*: – Changed fields "ID[7:6] / ID[3:2]" and "ID[5:4] / ID[1:0]" to "ID7_4_D3_0" – Changed field "ID[3:0]/DATA[15:12]" to "ID3_0_DATA15_12" In *Section 1.3.2.18, Channel 'n' Clock Control Register (n = 0 to (CH-1)) (CHn_CLK_CTRL)*, added note "The value provided..." to "CM_PRSC" field description *Section 1.3.2.20, Channel 'n' Configuration Register (n = 0 to (CH-1)) (CHn_CONFIG)*: in "FIL_CNT" field description, changed expression "This indicates the number" to "This indicates twice the number" In *Figure 28 (Enhanced Serial Message with 4-bit ID field)*, changed "Data [15:12]" to "Data Field [15:12]" Section 1.4.1, Initialization sequence: added note "When configuring the Single Edge Nibble Transmission (SENT)..." *Section 1.4.5, Adjustment for variation in sensor (Tx) clock*: added note "At ambient temperatures above 114 degree C..." In Section 1.4.6, Input programmable filter, added value of FIL_CNT in sentence "A timing diagram of the input filter when FIL_CNT is 4 …" Section 1.4.7.5, CRC check: added note "The Single Edge Nibble Transmitter (SENT) Receiver..." Section 1.4.7.6, Pause pulse diagnostic: substantial changes In *Section 1.4.8, Time stamp logic*, changed required prescaler value from 60 to 59 when the high frequency receiver clock is 60 MHz Chapter 50, LINFlexD Throughout chapter: reorganized bitmaps to change array of fields to single field for selected registers; Changed relationship for baud rate calculation from "LIN_CLK > PBRIDGEx_CLK ≥ 1/3*LIN_CLK" to "(2/3)*LIN_CLK > PBRIDGEx_CLK > 1/3*LIN_CLK". Updated *Figure 1 (Block diagram)* *Section 1.3.2.5.1, Header error*: in Case 2, changed "LINCR1 = 1" to "LINCR1 = 0" *Section 1.3.2.5.6, Timeout error*: added expression "+ LINFlexD_LINTCSR[CNT]" to the end of the bullet list item relative to Case 2 "At the end of the ID..." *Section 1.3.2.5.7, Noise*: changed paragraph "When OSR = 8, majority of 2nd, 3rd, and 4th samples..." to "When OSR = 8, majority of 4th, 5th, and 6th samples..." In *Section 1.3.2.5.6, Timeout error*, replaced "ocr_1" with "ocr_2" In *Figure 9 (Incomplete response (for example, missing checksum))*, and *Figure 10 (No response)*, replaced "OCR1" with "OCR2" In *Section 1.3.4.7, UART transmitter*, added notes "Invalid writes to BDRL which ..." and "When the FIFO is full, a read to..." *Table 19: LINCR1 field descriptions*: updated descriptions of fields "SLEEP" and "INIT" *Section 1.4.2.2, LIN Interrupt enable register (LINIER)*: renamed field relative to bit 28 to "DBEIE_TOIE" (was "DBIE/TOIE")

Table 1167. Revision history(Continued)

Date	Rev	Changes
27-Feb-2015	3 (cont'd)	*Section 1.4.2.3, LIN Status Register (LINSR)*: – Renamed field relative to bit 24 to "RXBUSY" (was "RXbusy") – Changed access type of field "AUTOSYNC_COMP" to "w1c" (was read-only) Updated *Section 1.4.2.5, UART Mode Control Register (UARTCR)* Updated *Section 1.4.2.6, UART Mode Status Register (UARTSR)* Removed Section "External signal description" *Table 41: 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 1.4.2.24, UART Current Timeout Register (UARTCTO)*: reset list item *UARTCTO becomes equal to UARTPTO* changed to *UARTCTO becomes equal to UARTPTO-1* In *Table 45: PTD field descriptions*, updated description of fields "IFD" and "EN" *Chapter 6: Reset Generation Module (MC_RGM)* *Table 15: 'Functional' Event Reset Disable (RGM_FERD) register field descriptions*: swapped fields "D_CWD" and "D_SWT" (bits 22 and 23) *Chapter 58: Boot Assist Flash (BAF)* Changed LINFlexD baud rate value to 250 kbaud with the IRCOSC as reference clock (was the external crystal oscillator frequency divided by 833.333) Moved sections "Search boot header and boot options", "Memory map", "LINFlexD_0 and MCAN_1 pin configuration" to *Section 6.7.9, BAF configuration* *Section 58.2.1, Boot modes*: in serial boot mode, – Changed life cycle from "CUST_DEL" to "IN_FIELD" – Added "M_CAN" interface *Section 58.2.9.3, Baud rates*: updated the division factor for baud rate calculation from 833.33 to 80 In *Section 58.2.8, Optionally perform a serial boot*, added paragraphs "The clock source in the MC_CGM is configured..." and "If CAN message not containing 8 bytes..." *Chapter 1: System Status and Configuration Module (SSCM)* *Section 1.4, Memory map and register definition*: added note to beginning of section, "All registers are accessible via 8-bit..." *Section 1.4.1.16, Life Cycle Status Register (LCSTAT)*: set bit 22–23 as Reserved (was field "PLC") *Table 25: LCSTAT field descriptions*: added "JDP_PROD = 110" to "LC" field description *Chapter 1: Power Management Controller digital interface (PMC_dig)* Section 54.3, Memory map and register definition: added register "EPR_VD2" *Figure 30 (Reset Event Enable register (REE_VD10))*: changed reset value of bit 19 to '0' *Table 50: EPR_TD field descriptions*: updated all field descriptions *55: Power Control Unit (MC_PCU)*: unchanged

Table 1167. Revision history(Continued)

Date	Rev	Changes
27-Feb-2015	3 (cont'd)	*Chapter 1: Mode Entry Module (MC_ME)*: no substantial changes *57: e200z215An3 Core Debug Support*: unchanged *Chapter 65: Buddy Device—Debug and Calibration Interface (DCI)* *Figure 1443 (DCI control register (DCI_CR))*: renamed field relative to bit 8 to "FPD_DEST" (was "FPD_DEST_RST") *Chapter 66: Buddy Device—JTAG Controller (JTAGC)* *Table 1529: 6-bit JTAG instructions*: removed "ACCESS_AUX_DWPU" row *Chapter 59: Buddy Device—JTAG Master (JTAGM)* *Table 1346: JTAGM_MCR register field descriptions*: updated binary combination descriptions for fields "EVTI0_ASSERT" and "EVTI0_ASSERT" Revised *Section 59.5.1.2, Status register (JTAGM_SR)* *Chapter 68: Password and Device Security Module (PASS)* *Table 887: LCSTAT register field descriptions*: added "110 JDP Production" to the "LIFE" field description *Table 897: Debug Interface Enable Truth Table*: – Removed the columns "Production Disable mode" and "Clock Jitter Enable" – Added JDP Production to "Customer Delivery" in column Lifecycle *Table 898: Flash Memory Read Protection Truth Table*: added Censorship "Censored" information for Lifecycle "Failure Analysis" *Chapter 86: Wakeup Unit (WKPU)* *Section 86.2.2, Features*: changed number of I/O external interrupts to 13 (was 16) Updated disclaimer
26-Sep-2017	4	"SPC572LxB" updated to "SPC572Lx" through the document *Chapter 2: Introduction* Replaced "eLQFP100" with "eTQFP100" *Chapter 3: Embedded memories* *Section 3.2.1: System SRAM*: Replaced "PFCR1" with "PRCR1" *Section 3.4: Security features*: Removed the note. *Chapter 4: Signal description* Replaced "eLQFP100" with "eTQFP100" *Chapter 5: Memory map* *Table 11: UTEST flash memory map*: – Updated the "DCF Records" row. – Updated all "OPACR" with "PCTL" *Chapter 6: Device configuration* *Table 12: Reset settings for e200z215An3 resources*: Value of "MMUCFG" replaced from "-" to "0x0000_0003"

Table 1167. Revision history(Continued)

Date	Rev	Changes
26-Sep-2017	4 (cont'd)	*Table 40: PIT1 memory map* renamed (was earlier titled PIT0 memory map) *Table 41: PIT0 memory map* renamed (was earlier titled PIT1 memory map) *Table 53: Boot header structure*: – Updated the table. – Removed table "Boot header configuration field description" *Chapter 7: Reset and Boot* *Section 7.2.2.1: Destructive resets*: Removed the bullet point "Reset escalation....". *Chapter 8: Device Configuration Format (DCF) Records* *Table 71: DCF client list*: Removed all rows of STCU. *Chapter 12: Core e200z215An3 description* *Figure 50: Machine State Register (MSR)*: Updated the figure. *Table 96: MSR field descriptions*: Updated the table with new fields. *Chapter 21: Clocking* *Figure 178: CMU0 block diagram*: Removed text "f_{CLKMN1} < f_{CLKMT0_RMN}/4" from "CLKMN1 Supervisor" block. *Chapter 22: PLL Digital Interface (PLLDIG)* *Section 22.5.2.3: PLL Divider Register (PLLDV)*: Removed text "RFDPHI, RFDPHI1). *Section 22.6.2: Clock configuration*: Updated the "VCO frequency" equation Removed all instances of PLL1. *Chapter 24: Clock Generation Module (MC_CGM)* *Figure 214: Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)*: Updated the reset value. *Chapter 28: Flash memory controller (PFLASH Controller)* *Section 28.4.1.4: PFlash Calibration Remap Control Register (PFCRCR)*: Updated the description for SAFE_CAL field. *Section 28.5.13.3: On-Chip Overlay RAM Feedback Check*: Removed text "If a mismatch...." *Chapter 29: Embedded Flash Memory (MP55)* *Section 29.3.2.13: Test Mode Disable Password Check register (TMD)*: Deleted text between brackets "Test Mode Disable Override Passcode at address offset 0x3DC8" *Table 321: TMD field descriptions*: Added Note "PWD bitfield is byte-reversed..." *Chapter 35: Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface* Reformatted *Figure 315: Module Configuration Register (MCR)*

Table 1167. Revision history(Continued)

Date	Rev	Changes
26-Sep-2017	4 (cont'd)	*Chapter 36: Successive Approximation Register Analog-to-Digital Converter (SARADC)* *Figure 362: Main Configuration Register (MCR)*: Bit 1 changed from "Reserved" to "EDCSELF". *Chapter 40: Software Watchdog Timer (SWT)* *Section 40.4: Functional description*: Removed line "The SWT_CR[CSL] bit selects ...drive the down counter" *Section 44: CAN Subsystem* *Section 44.3.1: Features*: Added footnote for CAN_FD "The CAN FD implemented...2015"" *Chapter 46: Deserial Serial Peripheral Interface (DSPI)* *Figure 600: DSPI Status Register (DSPI_SR)*: Bit "TXRXS" changed from "w1c" to "Read-only" *Chapter 51: Reset Generation Module (MC_RGM)* Removed *'Destructive' reset escalation* Removed *'Destructive' Reset Escalation Threshold (RGM_DRET) register* *Chapter 54: Power Management Controller digital interface (PMC_dig)* *Figure 870: Reset Event Enable register (REE_VD3)*: Changed the reset value of bits "LVD14_A" and "LVD14_IM" from 0 to 1. *Chapter Appendix A: Acronyms and abbreviations* Removed instance of STCU (Self test control unit)

Table 1167. Revision history(Continued)

Date	Rev	Changes
28-Feb-2019	5	Throughout document: Formatting and editorial changes. The following changes have been made: *Chapter 5: Memory map* *Table 11: UTEST flash memory map*: Added rows for Lifecycle Slot *Chapter 6: Device configuration* Updated *Section 6.7.2: Software Watchdog Timer (SWT) configuration* Updated row headed SWT3 *Table 36: Software Watchdog Timer (SWT) instances* Added *Section 6.7.2.1: Security watchdog* Updated *Section 6.9.1.2: Search boot header and boot options* “The BAF searches...KB blocks.” *Chapter 8: Device Configuration Format (DCF) Records* *Table 71: DCF client list*: – Removed all rows of TDR_LOCK – Added section BAF ‘Soft’ Clients and the subsequent rows *Section 8.4.2: Miscellaneous DCF registers*: Updated fields in *Figure 32: UTEST Miscellaneous DCF Client* and field descriptions in *Table 72: UTEST Miscellaneous DCF Client field descriptions* *Chapter 9: Power management* Removed Section: Checker core disabling *Chapter 28: Flash memory controller (PFLASH Controller)* *Section 28.4.1.4: PFlash Calibration Remap Control Register (PFCRCR)*: Updated bitfield SAFE_CAL *Chapter 29: Embedded Flash Memory (MP55)* *Table 292: Flash memory partition 0 memory map*: Updated the column headed LOCK and SEL *Table 293: Flash Memory Partition 1 memory map*: Updated the column headed LOCK and SEL *Chapter 31: Flash Memory Programming and Configuration* Updated all the examples *Chapter 36: Successive Approximation Register Analog-to-Digital Converter (SARADC)* Added information regarding the features external channel decode and external mapping Changed the name of the chapter from “Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface” to “Successive Approximation Register Analog-to-Digital Converter (SARADC)”

Table 1167. Revision history(Continued)

Date	Rev	Changes
28-Feb-2019	5 (cont'd)	*Chapter 47: LVDS Fast Asynchronous Serial Transmission – Interprocessor Communications* Changed the name of the chapter from "LVDS Fast Asynchronous Serial Transmission (LFAST) –Interprocessor Communications" to "LVDS Fast Asynchronous Serial Transmission – Interprocessor Communications" *Chapter 54: Power Management Controller digital interface (PMC_dig)* *Section 54.3.1.13: Reset Event Enable register (REE_VD14)*: Updated the default values for bitfield LVD14_A and LVD14_IM. *Chapter 56: Mode Entry Module (MC_ME)* *Table 937: MC_ME register description*: Updated row headed 0x0FE *Section 56.3.2.6: Invalid Mode Transition Status Register (ME_IMTS)*: A new bitfield S_MRIG had been added *Chapter 60: IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)* Removed Section: Abbreviations *Chapter 69: Wakeup Unit (WKPU)* *Table 1158: WKPU memory map* Updated the column headed 0x001C Added *Section 69.3.2.5: Wakeup Request Enable Register (WRER)* Updated *Figure 1230: External Interrupt Pad diagram*

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.

© 2019 STMicroelectronics – All rights reserved